تخطيطات الحاسوب

Computer Graphics

إعداد

سامي محمد شريف

أستاذ مشارك
قسم الهندسة الكهربائية والالكترونية
جامعة الخرطوم

يناير 2006

المحتويات

مقدمة

يعرف الحاسوب كنظام معلومات الالكتروني يعنى بقبول ومعالجة ونقل وعرض البيانات والتي تمثل معلومات معينة. التخطيط بالحاسوب احد تطبيقات الحاسوب ويعنى بالتقنيات الضرورية لقبول ومعالجة ونقل وعرض البيانات على نسق مرئي.

تقوم الحواسيب في الوقت الحالي بدور فعال كما تعتبر من الأدوات الأساسية في عمليات التصميم والرسم التخطيطي. يمكن اعتبار تخطيط الحاسوب كمجموعة من عتاد الحاسوب وبرامج التطبيقات التي تتوجه لهدف محدد وهو تكوين صورة. يقوم العتاد بدور حيوي في ضبط مقدرات نظم الحاسوب التخطيطية، حيث أنه يحدد درجة الاتصال الفعال بين المصمم والحاسوب.

مارس الإنسان منذ زمن عرض المعلومات على نسق مرئي، وقد يؤرخ لبداية لهذا النسق في العام 2000 قبل الميلاد حين استخدم الإسقاط المتعامد orthographic projection. في العام 1400 قبل الميلاد مثل ازدهار الحركة الفنية الرومانية دافع قوى لتتطور مجال التخطيط والرسومات. في العام 1600 قبل الميلاد كان لاستخدام نظام الإحداثيات وعلم الحسبان والفيزياء والضوء اثر كبير على هذا المجال. وكان راسم الإشارة والذي صنع في العام 1897 أول جهاز تخطيط الكتروني. خلال الأعوام 1950 وحتى 1970 استخدمت الحواسيب مع العرض المتجه بكثرة. وفى العام 1966 تم تصنيع أول عارض ماسح خطوط. تطور التخطيط بالحاسوب منذ ذلك التاريخ واليوم توجد كثير من تقنيات التخطيط بالحاسوب الفعالة.

تغطى تطبيقات الرسم بالحاسوب مجالات كثيرة مثل الأفلام والعاب الحاسوب والتصميم الصناعي والتصوير العلمي والتصوير الطبي والتصميم بواسطة الحاسوب ورابط المستخدم البياني. كان لكل هذه المجالات دور كبير في تطوير التخطيط بالحاسوب في الفترة الأخيرة.

العناصر الرئيسة للتخطيط بواسطة الحاسوب تتلخص في التالي:

- النمذجة: وتشتمل على
  - بدائيات التخطيط والتي ساعد على رسم النقاط والخطوط والمنحيات والأسطح الخ.
  - الصفات التي تحدد إشكال هذه الكمائن من الألوان وخواص الإضاءة
  - التحويل الهندسي والذي يستخدم لنقل ولف وإعادة حجم الكائن
- عوامل الإضاءة وتشتمل على
  - الرؤية، وهى العرض بكفاءة ووضوح
  - ومحاكاة انتشار الضوء من انعكاس وانكسار وانتشار الخ.
- الحركة
- التفاعل

التخطيط بالحاسوب مادة كبيرة ومتشعبة وقد تكمن الصعوبة في تحديد مدخل دراسة هذه المادة. فتعمد المادة على الرياضيات و تقنيات عتاد وبرامج الحاسوب وعمليات معالجة الإشارة الرقمية وغيره من مجالات المعرفة المرتبطة بدراسات الحاسوب. يتطلب دراسة هذا المنهج معرفة الدارس بصورة جيدة في مجالات الجبر الخطي والمصفوفات وحساب الملثمات والهندسة التحليلية. والحسبان.

يعتبر هذا المقرر مدخل لتخطيط الحاسوب وسنتعرف على أبدائيات الرسم الثنائية الأبعاد، و تحويلات وعمليات مشاهدة الكائن الثنائي الأبعاد وقص الكائن لحجم الشاشة المحدد وتمثيل وعمليات المشاهدة في الفضاء الثلاثي الأبعاد بالإضافة إلى بعض الموضوعات المتقدمة الأخرى. سيعطى المقرر أيضا الدارس خلفية كافية لكتابة تطبيقات تخطيط الحاسوب باستخدام مكتبة التخطيط OpenGL ولغة البرمجة C.

الوحدة الأولى: عتاد الرسم (التخطيط) بالحاسوب
Computer Graphics Hardware

1.1: مقدمة

يمكن تصنيف عتاد تخطيط الحاسوب بصورة عامة إلى أربع نظم فرعية وهى: الدخل والتخزين والإرسال والخرج. يتم أولاً تحويل معلومات المخطط إلى الحاسوب باستخدام أحد أجهزة الدخل المتخصصة المختلفة، وبعدها يتم تخزين هذه المعلومات في نسق معين يعتمد على نوعية وخواص الجهاز المستخدم وأخيرا يتم إرسال هذه المعلومات لجهاز الخرج المناسب للعرض أو المشاهدة.

يوجد عددت مئات من العوامل المتداخلة والمهمة للتخطيط بالحاسوب وقد تستخدم هذه العوامل المتداخلة لتميز عتاد نظم التخطيط بالحاسوب عن عتاد الحاسوب المعياري. ومن هذه العوامل التقنيات المختلفة لإنتاج المنظر ونوعيته، والحاجة للألوان والتمثيل الفضائي والاختلافات في نوعية الوسائط وسرعة جهاز الخرج الخ.

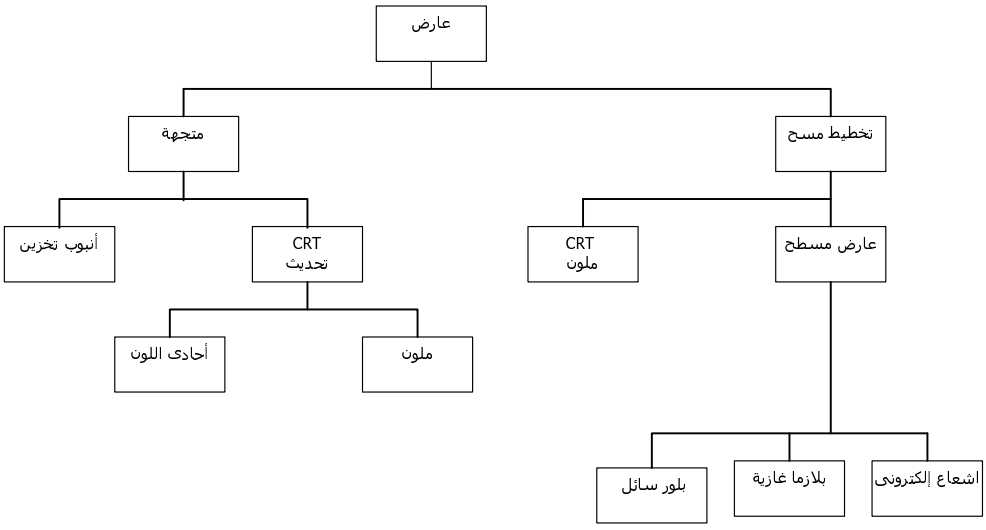
التطور في نظم المعالج الدقيق وعدم ارتفاع تكلفتها منذ نهاية الثمانيات من القرن الماضي أدى إلى تطور كبير في تقنيات التخطيط. شمل هذا التطور إلى تصميم نظما فرعية يمكن استخدامها كمسرعات للتخطيط، والسماح بتنفيذ دوال التخطيط الثلاثية الأبعاد على مستوى العتاد.

1.2. تقنية العرض ونظم التخطيط *display Technology and Graphics System*

تمثل طرفية العرض في نظم التخطيط الحاسوبية الرابط الأساسي بين المستخدم والنظام. تستخدم تقنيات مختلفة لتصميم وتصنيع أجهزة عرض التخطيط. يوضح الشكل 1.2 تصنيف أجهزة العرض المختلفة. كل من هذه الأجهزة لديها ميزات وقصور يجب أن تأخذ في الاعتبار عند اختيار عارض لتطبيق معين. ويحدد أداء العارض بالعوامل التالية: السعر و الإستبانة resolution والمقدرات التفاعلية.

الإستبانة تحدد مقدرة العارض على عرض التفاصيل. وتصل إستبانة النظم المستخدمة حاليا إلى حوالى 1280x1024 وتشير هذه الأرقام إلى عدد النقاط على الشاشة. العدد الكلى للألوان وعدد أللألوان التي يمكن عرضها في نفس الوقت أيضاً من العوامل الهامة التي يجب أخذها في الاعتبار. وتراوح عدد اللألوان التي يمكن عرضها في نفس الوقت على التوازي من 8 إلى 4096 لون من العدد الكلى للألوان التي قد تصل إلى 16 مليون. تشير التفاعلية على أمرين وهما

- المقدرة على إدارة جهاز دخل متفاعل
- مجال الحركية (مقدرة النظام لتوفير حركة مثيرة).

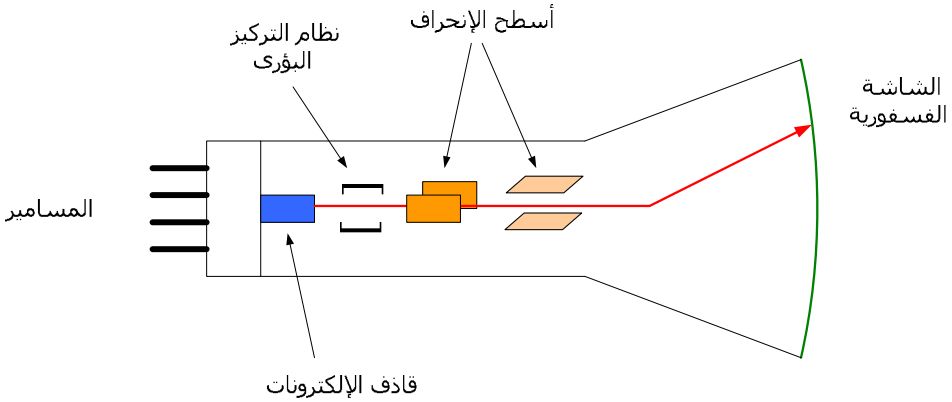


الشكل 1.1: التقنيات المستخدمة في تركيبات أجهزة عرض التخطيط

1.3. أنبوب الأشعّة المهبطية Cathode Ray Tube

يعتبر أنبوب الأشعة المهبطة أحد نظم العرض الأساسية والمستخدمة بكثرة في الوقت الحالي في نظم الحاسوب التخطيطية. يتكون هذا الأنبوب من أنبوب زجاجي مفرغ ينتهي بشاشة مخروطية الشكل. المكونات الأساسية لهذا الأنبوب هي (أنظر الشكل 1.2)

- القاعدة والتي تحتوى على مسامير التوصيل
- قاذف الإلكترونات
- نظام تركيز بؤري،
- أسطح الانحراف ألأفقي والرأسي
- شاشة فسفورية



الشكل 1.2: أنبوب الأشعة المهبطة

لإظهار المنظر يبعث قاذف الإلكترونات شعاع يمر عبر نظم الانحراف والتركيز البؤري قبل وصوله لموقع معين على الشاشة الفسفورية. نسبة لعدم وجود وزن للإلكترونات يمكن أعادة ضبط موقع هذا الشعاع بسرعة عالية.

تغطى شاشة أنبوب الأشعة المهبطة بمادة فسفورية تتوهج عندما تصاب بالشعاع الإلكتروني وتحدث نقطة إضاءة. وتصنع هذه المادة الفسفورية من مزيج من المواد مثل الكبريت والسيلكون والفضة. الضوء المبعوث من الفسفور يتلاشى بسرعة عالية عندما يتحول الشعاع من موقع لأخر. لمشاهد منظر مستمر على الشعاع الإلكتروني أعادة تخطيط (تحديث) الصورة على الشاشة على نفس الموقع.

الفترة الزمنية التي يكون الفسفور متهج تسمى استمرار ألأثر Persistenceوتساوى الزمن المطلوب للاضمحلال إلى عشر الشدة الأصلية. وتختلف قيمة هذا العامل لأنواع الفسفور المختلفة ويعتمد على مكونات الفسفور كما هو موضح بالجدول 1.2. ويلاحظ هنا أن العارض الذي يخزن المعلومات على الشاشة يستخدم فسفور لدية استمرار أثر أطول مثل P7 من الجدول. وفى حالة العارض الذي يحدث العرض بسرعة يستخدم فسفور لدية استمرار أثر قصير مثل P31 أو P4.

جدول 1.1: أنواع الفسفور المستخدمة في أجهزة العرض

نوع الفسفور	اللون	استمرار الأثر
P1	أصفر - أخضر	24 msec
P2	أصفر - أخضر	35-100 µsec
P4	أبيض	60 µsec
P7	أخضر/أبيض	300 µsec
P22-B	أزرق	22 µsec
P31	أخضر	40 µsec
P39	أصفر - أخضر	150 µsec

للمحافظة على شدة إضاءة جزء من الشاشة لفترة طويلة، يجب تنشيط أو تحديث الفسفور بصورة دورية بواسطة الشعاع الإلكتروني. قد ويؤدى عدم يحدث في أوقات مناسبة الفسفور إلى حالة تعرف بالارتعاش flicker تجعل المنظر يضمحل بين دورات التحديث على الشاشة.

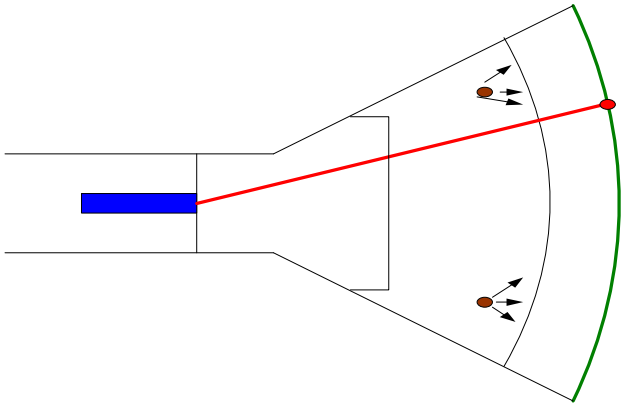
1.4. أنبوب التخزين Storage Tube

أنبوب التخزين وان كان لا يصنع في الوقت الحالي كان له الأثر الكبير في تطوير نظم التخطيط الحاسوبية. ويستطيع هذا الجهاز بالاحتفاظ بالمنظر على الشاشة دون حاجة للتحديث. يستخدم هذا الجهاز قاذفات مختلفة للالكترونيات وهى: (الشكل 1.3)

- قاذفات الفيضان flooding guns والتي تضرب الشاشة باستمرار بالإلكترونيات لتنشيط الفسفور في حده الأدنى
- قاذف كاتب الضربة stroke guns الواحدة والذي ينشط الفسفور إلى مستوى أعلى من الحد الأدنى

التشغيل المستمر لقاذفات الفيضان تجعل المنظر يستمر على الشاشة لوقت طويل دون تحديث. يتم مسح المنظر من الشاشة برفع الجهد في كل الشاشة والذي يحدث الإزالة.

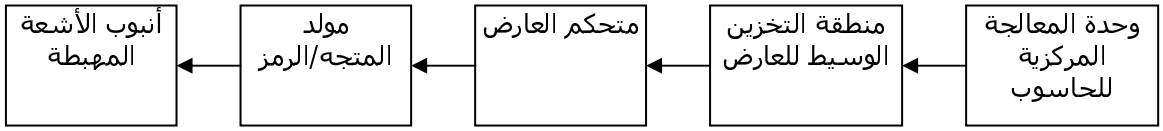
يعرض أنبوب التخزين كمية كبيرة من البيانات في نفس الوقت. وتوفر أستبانة قد تصل على 4096x4096. ومن أهم ميزات هذا النظام التكلفة المنخفضة الناتجة من عدم استخدام نظام تحديث إلكتروني معقد. يعانى هذا النظام من قصوراً في توفر مقدرات التلوين ومقدرته على إنتاج عرض ساكن فقط.



الشكل 1.3: أنبوب التخزين

1.5. عارضات تجديد المتجه Vector Refresh Displays

يولد عارض تحديث المتجه الصورة بتوجيه شعاع الإلكترونيات إلى نقاط عشوائية على الشاشة ويتم ربط هذه النقاط على نمط متجهات. يستخدم فسفور لديه استمرار أثر منخفضا في أنبوب الأشعة المهبطة. ولتجنب الارتعاش تجدد الشاشة مرات عديدة في الثانية. بالإضافة إلى أنبوب الأشعة المهبطة يحتاج هذا النظام إلى منطقة تخزين وسيطة، لحفظ المعلومات المطلوبة لتكوين الصورة ومتحكم للعارض الذي يرسل هذه المعلومات إلى مولد المتجه الذي بدوره يحولها إلى موقع الشعاع الإلكتروني على الشاشة. تعتمد هذه العملية على سعة الخازنة الوسيطة وسرعة المتحكم. الشكل 1.5 يوضح احد التشكيلات الممكنة لهذه العملية.



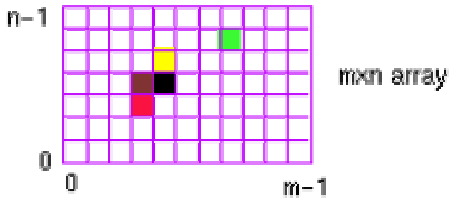
الشكل 1.5: نظام تحديث المتجه

يسمح عارض تجديد المتجه بتعديل إي من عناصر الصورة في الذاكرة الوسيطة، وحيث أن الفسفور المستخدم لديه استمرار أثر قصير يمكن استخدام دورة التحديث لتعديل المنظر الأصلي. هذه الخاصية تجعل هذا العارض مناسبا في تطبيقات ذات الحركة المستمرة. ينتج عارض تحديث المتجه صورة زاهية وواضحة ذات استبانه عالية. تعانى هذا العارض من التكلفة العالية وعدم المقدرة على عرض المناطق الكثيرة الألوان. استخدم عارض تجديد المتجه حتى منتصف الثمانيات من القرن الماضي عندما بداء بفضل عليها عارض خطوط المسح.

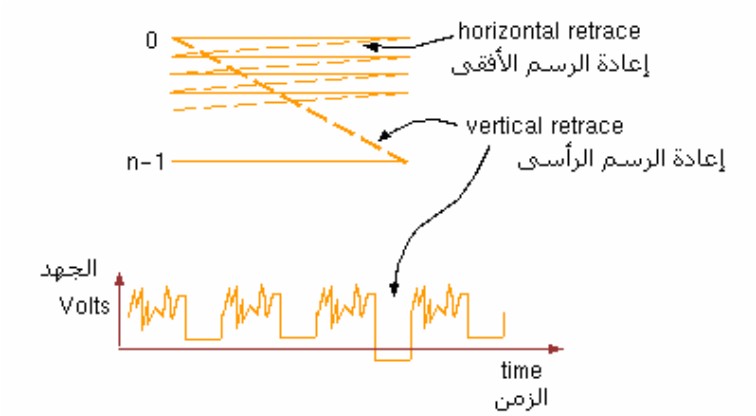
1.6. عارضات خُطوط المسح *Raster Scan Display*

استخدام عارضات خطوط المسح والمعتمدة على تقنيات التلفزيون في بداية السبعينات من القرن الماضي، حيث أعطت مجال التخطيط الحاسوبي دفعة قوية. وخلافا للنظم العرض التي تم التعرض لها سابقا والتي تصنف كأجهزة رسم خطوط يصنف عارض خطوط المسح كجهاز راسم للنقاط.

تقسم شاشة عارض خطوط المسح إلى مصفوفة xy من النقاط، تعرف كل منها بعنصر الصورة pixel وتتكون الصورة من عناصر الصورة هذه بجعلها نشطة أو خاملة، انظر الشكل 1.6. يتحرك شعاع الإلكترونيات في أنبوب الأشعة المهبطة أفقيا بدء من الركن الأيسر في أعلى الشاشة ويستمر من الشمال إلى اليمين في مستويات رأسية أفقية (الشكل 1.7). وتعاد هذه العملية حتى يتم تغطية كل الشاشة. عندها يعود الشعاع إلى الركن الأيسر أعلى الشاشة ويعاد المسح. يكون المنظر على الشاشة بتغير شدة الإشعاع إثناء تنقله. إن كان الموقع جزء من الصورة يمر الشعاع عليه بالشدة القصوى ويعود إلى الحد الأدنى من الشدة إن كان الموقع مظلماً.



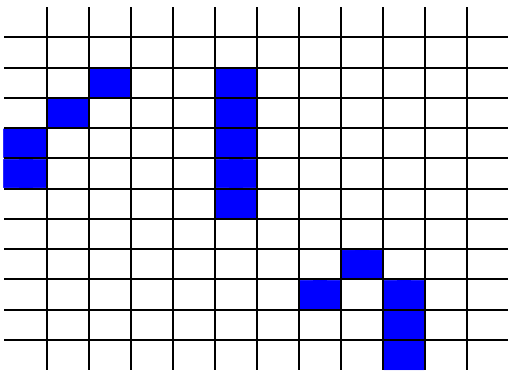
الشكل 1.6: مصفوفة عارض خطوط المسح



الشكل 1.7: أشارات المرئي في عارض ماسح الخطوط

تتمثل محدودية هذا العارض في إن كل عنصر صورة يجب التعامل معه منفرداً. خلال تحرك الشعاع عبر العارض يستلم معلومات عن مستوى الشدة المطلوبة في النقاط المختلفة لتكوين الصورة. يخزن مستوى التضمين لكل عنصر صورة في ذاكرة رقمية تعرف باسم الخازنة الوسيطة للتجديد أو الأطر (الشكل 1.8). عند تحرك الشعاع يستلم المتحكم البيانات من هذه الذاكرة ويحولها للجهد تماثلي ثم ترسل إلى مضخم شدة الشعاع. كل عنصر صورة يحتاج لموقع خاص به في الذاكرة مما يتطلب حجم ذاكرة كبير لنظام العرض هذا.

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0



الثنائيات في الذاكرة (خازنة التجديد الوسيطة) عناصر الصورة

الشكل 1.8: تكوين خازنة الأطر الوسيطة

تمتاز عارضات خطوط المسح بالمقدرة على تكوين صورة زاهية واضحة المعالم وخاصة عندما تستخدم الألوان. وتطورت استبانه هذه النظم في السنوات الأخيرة بصورة كبيرة مما جعلها تقترب من أداء عارضات تجديد المتجه. لا تتأثر عارضات خطوط المسح بتعقيدات الصورة ويمكنها عرض الصور المتحركة.

1.7. عارضات خُطوط المسح الملونة *Color Raster Display*

تعمل عارضات خطوط المسح الملونة بصورة مشابهة لعارضات اللون الأحادي monochrome، وتختلف عنها باستخدام مجموعة من الألوان الفسفورية لتكوين المنظر. تستخدم ثلاث شعاعان لإضاءة ثلاث أنواع مختلفة من الفسفور وهى الأحمر والأخضر والأزرق RGB. وبتغير شدة هذه أللألوان الأساسية يمكن إنتاج تتدرج كبير للألوان.

عادة تستخدم ثلاث قاذفات الكترونيات في عارضات خطوط المسح الملونة وتنظم على هيئة مثلث. وتنظم النقاط الحمراء والخضراء والزرقاء على نفس الهيئة. تضع شبكة معدنية مثقبة بين قاذفات الإلكترونيات والفسفور تعرف باسم ستار (قناع) الظل.، بحيث يسمح لكل قاذف بإثارة الفسفورية المقابل له. إن الثقوب الموجودة في الستار المعدني لا تسمح للشعاع بالسقوط في أكثر من نقطة فسفورية واحدة. شدة الشعاع تنتج مستويات مختلفة لشدة اللون الأساسي ما يعطى تتدرج كبيراً للأوان.

لعرض أللألوان على أنبوب الأشعة المهبطة يجب تعديل خازنة الأطر الوسيطة. كما ذكر سابقا إن كل عنصر صورة لديه موقع في الذاكرة في النظام الأحادي اللون يحتوى هذا الموقع على الثنائية 0 أو 1 تحدد شدة الشعاع الساقط على عنصر الصورة ويعرف هذا بسطح الثنائية bit surface. لعرض أللألوان تستخدم أسطح ثنائيات إضافية. استخدام N أسطح ثنائيات تعطى عدد من مستويات الشدة أو عدد ألوان تساوى N^2 يمكن عرضها في نفس الوقت. الجدول 1.2 ويوضح العلاقة بين عدد أسطح الثنائيات وعدد الألوان الممكنة.

جدول 1.2: أسطح الثنائيات

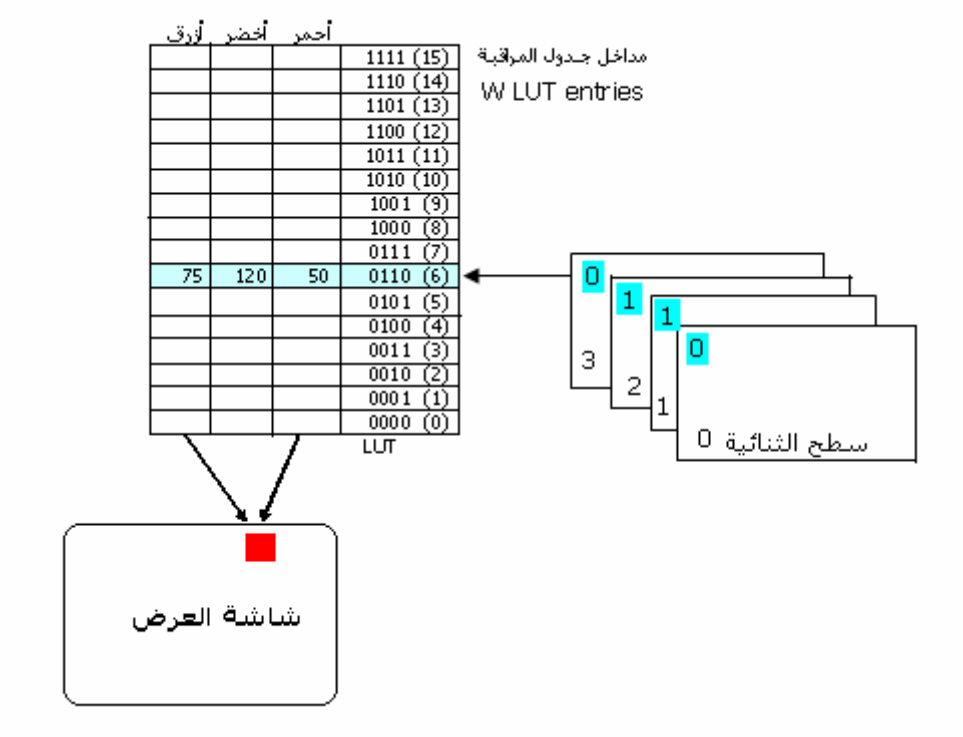
الألوان	ثنائيه لكل عنصر صورة
4	2
8	3
16	4
256	8
1024	10
4096	12
≈ 16 مليون	24

الحد الأدنى لعدد لأسطح الثنائيات الضروري لتكوين تخطيط ملون هو ثلاثة أسطح واحد لكل من الألوان الرئيسية الأحمر والأخضر والأزرق. البيانات المخزنة في هذه الأسطح تستخدم لتنشيط قاذفات الالكترونات الثلاثة في العارض الملون. يحدد لأي عنصر صورة على الشاشة ثلاث ثنائيات لتحديد مستوى الشدة، مما يعنى أن عنصر الصورة يمكن أن يأخذ واحد من ثمانية ألوان مختلفة كما هو موضح في الجدول 2.3.

جدول 1.3: الألوان والثنائيات

	أحمر	أخضر	أزرق
أسود	0	0	0
أحمر	1	0	0
أخضر	0	1	0
أزرق	0	0	1
أصفر	1	1	0
Cyan	0	1	1
Magenta	1	0	1
أبيض	1	1	1

استخدام عدد أضافي من أسطح الذاكرة (أسطح الثنائيات) يعطي مجال أكبر للألوان وشدة عناصر الصورة، ومن الضروري أعطاء المستخدم بعض المرونة في اختيار الألوان التي يريد إن يستخدمها. أحد الطرق لتحيق هذه المرونة باستخدام جدول المراقبة lookup table. عنوان الذاكرة في جدول المراقبة يحتوى على العدد المتوسع الذي يمكن ووضعه على إحداثيات عنصر الصورة في العارض كما هو موضح في الشكل 1.9. تحول أسطح الثنائيات إلى خازنة الأطر الوسيطة والرقم النتاج يتوافق مع الموقع المبرمج في جدول المراقبة. وفى هذه الحالة لا يستخدم قيمة عنصر الصورة للتحكم في الشعاع الالكتروني مباشرة بل يستخدم كمؤشر في جدول المراقبة. إن كان عدد أسطح الثنائيات يساوى N يجب أن يكون هناك 2^N مدخل في جدول المراقبة.



الشكل 1.8: صاد الأطر مع جدول المراقبة

1.8. عارضات اللوحات المستوية *Flat Panel Displays*

أكثر العارضات استخداما في الوقت الحالي هي أنبوب الأشعة المهبطة وهذه العارضات عادة ما تكون ضخمة وكبيرة الحجم ومحدودة المقاس، وذلك زاد الاهتمام بعارضات اللوحة المستوية. أكثر عارض اللوحة المستوية استخداما حاليا هي عارض البلازما Plasma و عارضة الاستضاءة الكهربائية electroluminescent وعارضات البلور السائل liquid crystal.

تتكون عارضة البلازما من صف من بصيلات نيون صغيرة التي يمكن تنشيطها أو عدم تنشيطها وتبقى على حالتها هذه حتى يتم أمرها بتغير الحالة. أي نقطة يمكن تنشيطها أو عدم تنشيطها دون الحاجة لتجديد كل الشاشة. تتكون اللوحة من ثلاثة أسطح زجاجية مع وجود بصيلات النيون في السطح الزجاجي في الوسط وخطوط التغذية الكهربائية الرأسية والأفقية على الأسطح الأخرى. تنشط إي من البصيلات النيون بعنونة المصفوفة وضبط الجهد في خطوط التغذية المقابلة لها في السطح الأمامي والخلفي. تعانى عارضات البلازما من محدودية الأستبانة والتكلفة العالية وبالمقابل تمتاز بأنها متينة جداً وتعمل دون الحاجة لدورات تجديد.

1.9. أجهزة الدخل Input Devices

توجد كثير من أجهزة الدخل للتخطيط بالحاسوب مثل الفارة mouse و عمود الإدارة Joystick و صندوق الأزرار bottom box والقلم الضوئي light pen الخ. وسنتعرض هنا فقط للفارة.

الفارة ببساطة جهاز يمد الحاسوب بثلاثة ثمانيات bytes من المعلومات (كحد ادني) تمثل

- مسافة الحركة في اتجاه X
- مسافة الحركة في اتجاه Y
- حالة الزر

ويتم توفير علامة مرئية على الشاشة بواسطة برنامج. ينتج حدث الفارة عندما يتغير شي ما مثل التحرك أو الضغط على الزر. ترسل الفارة إشارة إلى نظام التشغيل لأعلامه بالحدث المعنى ثم ترسل المعلومات الجديدة. يقوم نظام التشغيل عندها بتحديث موقع العلامة المرئية على الشاشة. ويقوم بإعلام برامج التطبيقات بالحدث المتعلق به. يستلم برامج التطبيقات هذه المعلومات بما يعرف بعملية المناداة العكسية callback procedure أو حلقة الحدث event loop. أدناه مثال بسيط لعملية المناداة العكسية

```
while (executing) do
{       if (menu event) ProcessMenuRequest();
        if (mouse event)
        {       GetMouseCoordinates();
                GetMouseButtons();
                PerformMouseProcess();
        }
        if (window resize event) RedrawGraphics();
}
```

تستخدم الفارة عادة لتحقيق أساليب الدخل على النحو التالي:

- محدد الموقع Locator
  يحدد الموقع على الشاشة باستخدام علامة مرئية
- رباط مطاط Rubber Band
  يضبط حجم وموقع كائن التخطيط

1.10. تمارين

1. تفحص عتاد التخطيط بالحاسوب الخاص بك، أعطى قائمة بكل المواصفات الفنية لآجهزة الدخل والخرج.
2. فارن ميزات وقصور كل تقنيات العرض الرئيسة
3. أشرح كيف يمكن لبعض عارضات مسح الخط عرض 256 لون في نفس اللحظة من مجموع الألوان الكلى والذي يساوى 16.7 مليون.

الوحدة الثانية: الدخل والخرج وبرمجيات التخطيط
Output and Input and Graphics Software

2.1. تبعية الجهاز *Device Dependence*

ذكر في الوحدة السابقة بان العناصر الرئيسة لعتاد نظام التخطيط بالحاسوب هي:

- المعالج processor
- الذاكرة memory
- صاد الأطر framebuffer
- أجهزة الخرج
  - العارض CRT, LCD
  - الطابعة
- أجهزة الدخل
  - لوحة المفاتيح، الفارة ، الخ.

برمجة نظم التخطيط بالحاسوب تتطلب أن تكون الأجهزة غير تابعة، بمعنى أخر، يجب أن نكتب البرنامج بالبعد قدر المستطاع من صفات عتاد معين.

أكثر أجهزة خرج التخطيط استخداما في مجال الحاسوب تتدرج تحت الأنواع التالية:

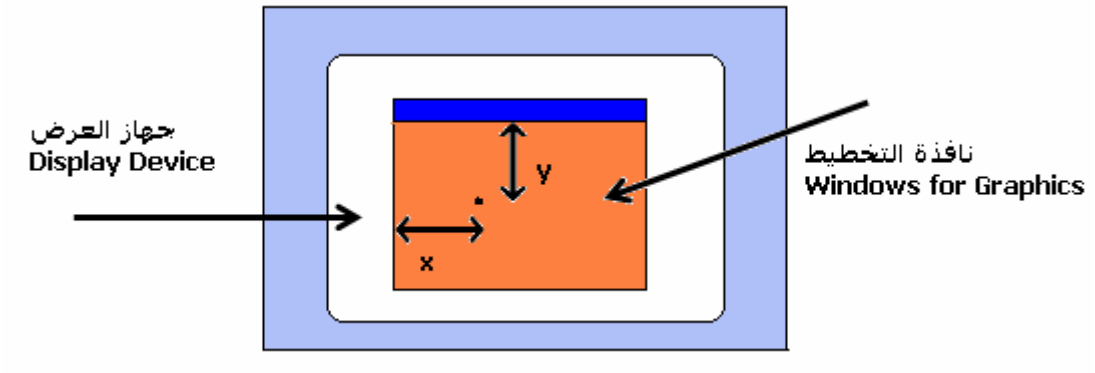
- شبكة خطوط المسح *Raster* type
- ورسم بياني النقاط plot points

في هذه الأجهزة تضع كل النقاط dot (عناصر الصورة pixel) التي تكون الصورة مباشرة في ذاكرة وصول عشوائي random access memory . وتصل وحدة المعالجة المركزية CPU إلى هذه الذاكرة مباشرة أو عبر مسجلات تحكم control registers

إن عدد الثنائيات bits الذاكرة المستخدمة لكل عنصر صورة يحدد مجال الشدة intensities التي يمكن استخدمها. فمثلا ثنائية واحدة تعنى أن عنصر الصورة ابيض أو أسود وثمانية ثنائيات تسمح 255 شدة (أو لون) مختلف لعنصر الصورة. حاليا من العادة استخدام 24 أو 32 ثنائية لتمثيل عنصر الصورة وبسمح هذا التمثيل بملايين الألوان (انظر الوحدة الأولى).

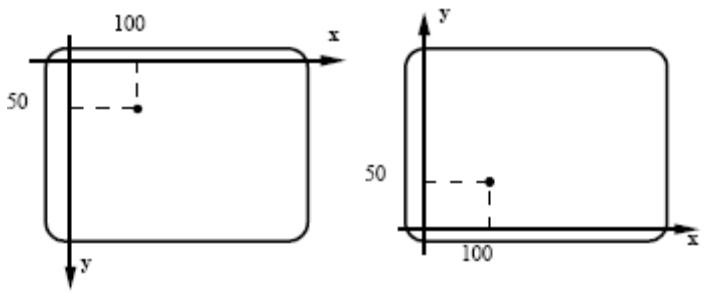
تسمى عدد عناصر الصورة في الشاشة الإستبانة (درجة الوضوح) resolution وتعطى عادة بدلالة مكونات x و y [XRes,YRes]، وفى حالة الطابعات الليزرية تحدد الإستبانة بعدد النقاط في كل بوصة.

أكثر أجهزة عرض التخطيط استخداما أجهزة شبكة خطوط المسح حيث يمكننا استخدام إجراء procedure مثل SetPixel(x,y,colour) لتغير لون عنصر صورة معين. الإحداثيات المستخدمة تمثل العنوان الحقيقي لعنصر الصورة. (أنظر الشكل 2.1)



الشكل 1.2: معنى الإجراء SetPixel(x,y,colour)

تتمثل مشكلة استخدام الإحداثيات الفعلية لعنصر الصورة في أن أجهزة العرض المختلفة قد تستخدم أساليب مختلفة لتحديد عنوان عنصر الصورة على الشاشة. ففي بعض هذه الأجهزة تكون نقطة تقاطع محاور الإحداثيات origin في الركن الشمالي الأعلى وفى بعض الأجهزة الأخرى تكون هذه النقطة في الركن الأسفل على يمين الشاشة كما هو موضح في الشكل 2.2.



الشكل 2.2 تغير أسلوب عنونة عنصر الصورة من نظام إلى أخر

يوفر لنا نظام التشغيل إمكانيات رسم التخطيط على مستوى عنصر الصورة. فمثلا في رابط المبرمج البيني Windows 32 يمكن استخدام الإجراءات التالية:

```
MoveToEx(hdc xpix, ypix);
LineTo(hdc, xpix, ypix);
TextOut(hdc, xpix, ypix, message, length);
```

حيث hdc معين لنوع النافذة و xpix و ypix نظام الإحداثيات.

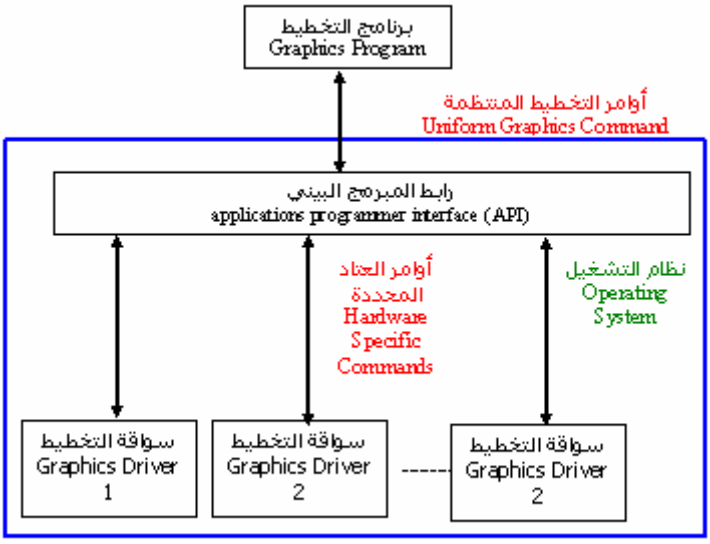
ونوضح هنا إن رابط المبرمج البيني لا يوفر لنا عدم تبعية مطلقة للجهاز، فمثلا إن كتبنا برنامج بحيث تحتوى كل الكميات في عنصر الصورة، بالتالي لا يحدث تغير إن غيرنا حجم النافذة.

إن الحاجة لكتابة برامج تخطيط غير معتمدة على جهاز العرض تتمثل في التالي:

- في التطبيقات العادية هناك حاجة لضبط حجم الصورة إن تغير حجم النافذة
- يجب أن تكون الصورة غير معتمدة على الإستبانة
- السماح بنقل الصورة بين نظم مختلفة (حواسيب شخصية ومحطات عمل وحواسيب فائقة القدرة الخ)

2.2. رابط المبرمج البيني *applications programmer interface (API)*

في أغلب ألأحيان لا نهتم كثراً بتفاصيل الجهاز، حيث يقوم نظام التشغيل operating system بالإهتمام ذلك بواسطة ما نسميه رابط المبرمج البيني.applications Programmer interface or API. والذي يوفر للمبرمج مجموعة من الإجراءات المنتظمة لرسم الصورة بغض النظر عن الجهاز المستخدم (أنظر الشكل 2.3)



الشكل 2.3: رابط المبرمج البيني

أكثر روابط المبرمج البينية استخداما هي

- OpenGL
- Direct3D
- Java3D
- VRML
- Win32 API
- GKS
- PHIGS

سنتعرض في هذا المقرر لأمثلة لاستخدام مكتبة التخطيط أما في بقية هذه الوحدة سنتعرف على بعض الأمثلة من مكتبة التخطيط GKS و PHIGS بالإضافة إلى OpenGL. إن مكتبة التخطيط OpenGL لديها الخواص التالية:

- لا يعتمد على العتاد حيث يمكن استخدامه على الحواسيب الشخصية PC ومحطات العمل workstations والحواسيب الفائقة القدرة Supercomputer
- يعتمد على نظام التشغيل ويدعم نظم التشغيل التالية: Windows NT, Windows 2000, Windows XP Linux and Unix
- يمكن أن ينفذ باستخدام البرامج مثال المعالج أو العتاد مثل لوحة تسريع التخطيط accelerated graphics card إن وجدت.
- يمكن استخدامه مع لغات البرمجة التالية: C, C++, Ada, Fortran, Java
- يدعم أداء المضلع polygon rendering التخطيط البنيوي texture mapping و وعدم استخدام الاسم المستعار anti-aliasing
- لا يدعم متابعة الإشعاع ray tracing و أداء الحجم volume rendering

لكتابة برنامج متنقل لا يعتمد على جهاز التخطيط علينا إزالة التبعية للجهاز الموضحة أعلاه من اغلب برامجنا للتخطيط. في بعض التطبيقات مثل ألعاب الحاسوب computer games، نحتاج لاستغلال كل مساحة الشاشة (العارض)، وفى مثل هذه الحالة نحتاج أن يتواءم مع التغيرات في الإستبانة.

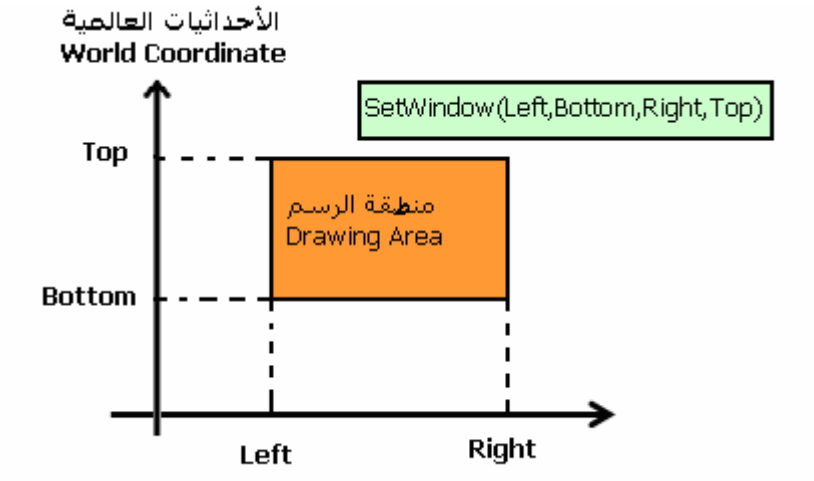
ومن كل مذكر أعلاه نخلص إلى انه لا يمكننا كتابة برنامج بكفاءة باستخدام عنوان عنصر الصورة.

2.3. نظام الإحداثيات العالمي *World Coordinate System*

تتطلب تطبيقات التخطيط الفعلية أن كل شي يرسم في النافذة أن يكون غير معتمد على موقع النافذة على الشاشة وحجمها. إن استخدام نظام الإحداثيات العالمي يحقق عدم الاعتمادية هذه. حيث تحدد مساحة الرسم باستخدام وحدات مناسبة للتطبيق المعنى مثل المتر والسنة الضوئية والميكرون الخ. وفى العادة يستخدم الآمر التالي لتحديد الإحداثيات العالمية

SetWindow(left,bottom,right,top)

يمكن اعتبار هذا كنافذة في العالم تتوافق مع النافذة في الشاشة (أنظر الشكل 2.4)



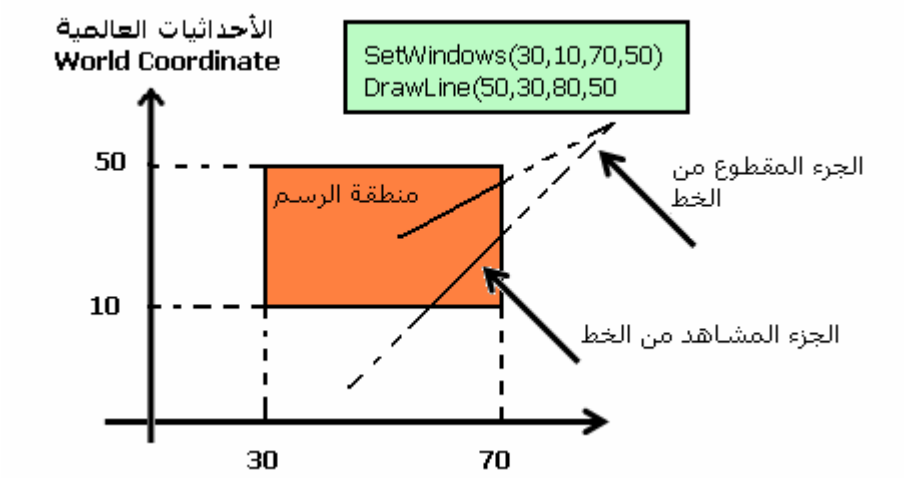
الشكل 2.4: نظام العالمي للإحداثيات

بعد تحديدنا لنظام الإحداثيات العالمي يمكننا تحقيق بدائيات الرسم التي تستخدم معه مثل

```
DrawLine(x1,y1,x2,y2);
DrawCircle(x1,y1,r);
DrawPolygon(PointArray);
DrawText(x1,y1,"A Message");
```

في العادة اى جزء من كائن التخطيط خارج النافذة يتم قصه. وهذا قد يمثل مشكلة في بعض الأحيان. فمثلا في الشكل 2.5 يعطى التخطيط المتحصل عليه باستخدام التعليمات التالية:

```
SetWindow(30,10,70,50);
DrawLine(50,30,80,50);
DrawLine(80,50,50,5);
```

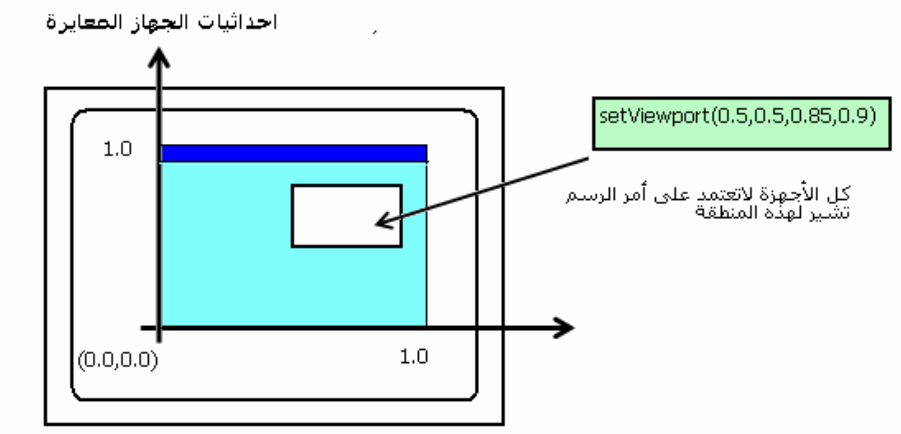


الشكل 2.5: قص جزء من الرسم خارج النافذة

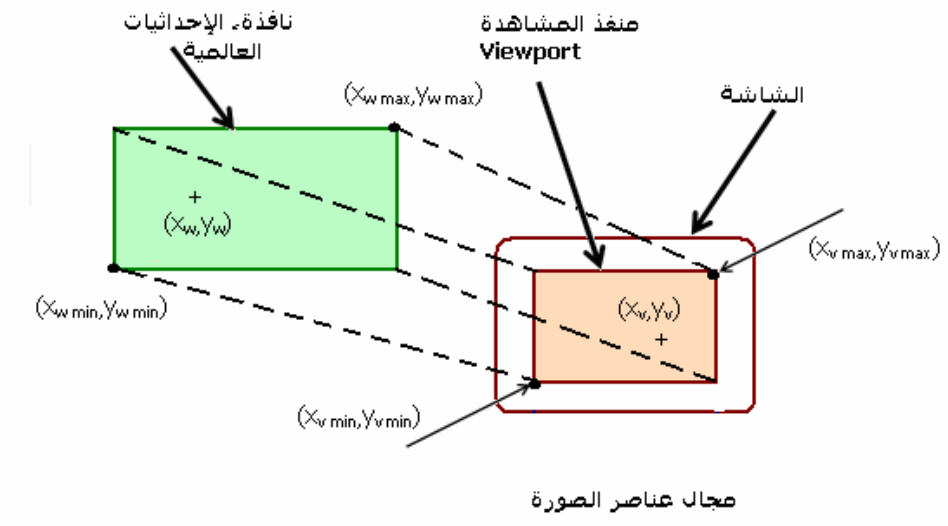
تتجنب بدائيات الرسم الغير معتمدة على جهاز العرض استخدام مجموعة عوامل شاملة، وتستخدم عدد من الصفات attributes لتحديد الكائن. فمثلا الخط يحدد بالصفات التالية: ألأسلوب(مستمر أو متقطع) و السمك(نقاط) و اللون, أما النص text فيحدد باستخدام الحروف والحجم و اللون

2.4. المعيارية *Normalization*

حتى نتمكن من مشاهدة المخطط على الشاشة لابد من ربط بدائيات التخطيط الغير معتمدة على الجهاز مع أوامر الرسم المعتمدة على الجهاز ويتم هذا الربط باستخدام عملية المعايرة. (انظر الشكل 2.6)



الشكل 2.6: المعايرة



الشكل 2.7: العلاقة بين إحداثيات النافذة ومنفذ المشاهدة

نحتاج في المعايرة إلى ترجمة الإحداثيات العالمية إلى مجموعة إحداثيات مناسبة للرسم باستخدام رابط المبرمج البيني. فلابد في البداية من منادة رابط المبرمج البيني حتى نتمكن من تحديد عناوين عناصر الصورة لأركان المساحة التي سنستخدمها من نظام التشغيل. باعتبار الشكل 2.7 يتم الربط بين نظام الإحداثيات العالمي ونظام إحداثيات جهاز العرض باستخدام العلاقات التالية :

$$\frac{x_v - x_{v\min}}{x_{v\max} - x_{v\min}} = \frac{x_w - x_{w\min}}{x_{w\max} - x_{w\min}} \qquad 3.1a$$

$$\frac{y_v - y_{v\min}}{y_{v\max} - y_{v\min}} = \frac{y_w - y_{w\min}}{y_{w\max} - y_{w\min}} \qquad 3.1b$$

اعد ترتيب هذه المعادلة تتحصل على

$$x_v = (x_w - x_{w\min})\left(\frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}}\right) + x_{v\min} \qquad 3.2a$$

$$y_v = (y_w - y_{w\min})\left(\frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}}\right) + y_{v\min} \qquad 3.2b$$

الحدود

$S_x = \left(\frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}}\right)$ و $S_y = \left(\frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}}\right)$ ثابتة constant لكل النقاط وتمثل ببساطة عامل ضبط مقاس في اتجاه x و y. إذا كان $S_x \neq S_y$ يحدث تشويش distortion في الصورة. مفهوم نسبة الواجهة aspect ratio يشير إلى هذه الحالة، حيث تعطى نسبة الواجهة بقسمة العرض على الارتفاع

$$AR = \frac{x_{mav} - x_{\min}}{y_{\max} - y_{\min}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.3$$

إذا كانت نسبة الواجهة لكل من النافذة ومنفذ المشاهدة تتساوى عليه يكون $S_x=S_y$ و يحث تشويش للصورة.

ويمكن دمج المعادلات أعلاه في المعادلات الخطية البسيطة التالية:

$$x_v := x_w * A + B;$$
$$y_v := y_w * C + D;$$

تمثل منافذ المنظر viewports اصغر جزء من النافذة يمكن عرضه. وعند اختيارنا لمنفذ منظر، إن الأسلوب المتعارف علية لنقل (تخطيط) الإحداثيات العالمية إلى المنفذ وليس لكل مساحة الرسم. تعرف منافذ المنظر في إحداثيات الجهاز المعيارية ويكون لكل نافذة الرسم إحداثيات أركان [0.0,0.0] و [1.0,1.0]

إن استخدام منافذ المنظر يغير من طريقتنا للمعايرة، حيث نحتاج في هذه الحالة للتالي:

- منادية رابط المبرمج البيني من نظام التشغيل لتحديد عناوين عناصر الصورة في أركان النافذة
- استخدام محيط منفذ المنظر لحساب عناوين عناصر الصورة للمساحة التي سيظهر عليها الرسم
- حساب عوامل المعايرة A و B و C و D.

مثال

أوجد مصفوفة التحويل التي تنقل النقاط في النافذة التي لديها الركن الأسفل الأيسر عند (2,2) والركن الأعلى ألأيمن عند (6,5) إلى منفذ مشاهدة معاير لديه الركن الأسفل الأيسر عند (0.5,0.5) والركن الأعلى ألأيمن عند (1,1).

الحل:

عوامل النافذة ومنفذ المشاهدة:

$x_{wmin} = 2$	$x_{vmin} = 0.5$
$x_{wmax} = 6$	$x_{vmax} = 1$
$y_{wmin} = 2$	$x_{vmin} = 0.5$
$y_{wmax} = 5$	$y_{vmax} = 1$

باستخدام المعادلة 3.2 تتحصل على

$$S_x = \left(\frac{1-0.5}{6-2}\right) = \frac{1}{8}$$
$$S_y = \left(\frac{1-0.5}{5-2}\right) = \frac{1}{6}$$

عليه تكتب المعادلة 3.2 على الهيئة التالية:

$$x_v = (x_w - x_{w\min}) + x_{v\min}$$
$$y_v = (y_w - y_{w\min}) + y_{v\min}$$

في هيئة المصفوفة

$$[x_v \quad y_v \quad 1] = [x_w \quad y_w \quad 1]\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ (-S_x.x_{w\min} + x_{v\min}) & (-S_y.y_{w\min} + y_{v\min}) & 1 \end{bmatrix}$$

بالتعويض تتحصل على مصفوفة النقل التالية:

$$\begin{bmatrix} \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{6} & 0 \\ \frac{1}{4} & \frac{1}{6} & 1 \end{bmatrix}$$

2.5. دوال الخرج Output Functions

كل معايير التخطيط تعطى متواليات من بدائيات الخرج output primitives، يمكن باستخدامها إنتاج مشاهد معقدة. هذه البدائيات قد تكون خط linel أو واسمة marker ثائي أو ثلاثي الأبعاد. ، الجدول 2.1 يوضح بعض البدائيات في مكتبة PHIGS.

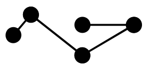
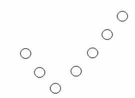
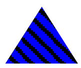
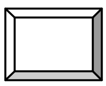
تدعم مكتبة التخطيط مبرمج التطبيقات بدوال خرج ابتدائية مثل:

DRAW_POLYINE(SetOfPoints)

ترسم هذه الابتدائية مقطع خط يصل مجموعة النقاط المحددة.

توفر الابتدائية معلومات عن الموقع والهندسة، لتحديد عوامل أخرى مثل اللون ونوع الخط الخ، عرفت صفات رسم drawing attributes متصلة بكل ابتدائية عبر دالة SET <attribute>. الجدول 2.2 يوضح صفات الابتدائية polyline.

الجدول 2.1. ابتدائيات خرج PHIGS

الابتدائية	الوصف	مثال
polyline	متسلسلة نقاط تحدد مجموعة من الخطوط المتصلة	
Polymarker	رموز تضع فى مواقع محددة	
Text	رموز متصلة تضع في أماكن محددة	**graphics - تخطيط**
Fill area	تعبئة المساحة الداخلية للمضلع ولا تشمل الحافات بأنماط مختلفة.	
Fill area set	تعبئة مساحات مضلعات معقدة وقد تشمل الحافات بأنماط مختلفة	

الجدول 2.2. صفات الابتدائية polyline

الصفة atributte	الوصف
Linetype	للاختيار من: صلب: مقطع: مركز:
Linewidth	اختيار سمك الخط كنقاط متعددة من خط العارض
Polyline Color Index	اختيار لون الخط من جدول ألوان المحطة.

2.6. دوال الدخل Input Functions

عدم الاعتمادية على الجهاز تتطلب استخدام البرمجيات لأنواع مختلفة من ألأجهزة بأقل حد ممكن من التعديلات. لتحقيق هذا تستخدم أجهزة الدخل المنطقية logical input devices أو دوال الدخل input functions لتمثل أجهزة الدخل التي تتصف بنفس الصفات. أجهزة الدخل المنطقية تصنف إلى ست مجموعات اعتمادا على قيم البيانات التي يرجعها الجهاز لبرنامج التطبيقات. هذه المجموعات موضحة في الجدول 2.3. كل من هذه الأجهزة المنطقية يمكنها استخدام أكثر من جهاز فعلى مرتبط بها.

الجدول 2.3. تصنيفات الدخل المنطقي

الصنف	الوصف
Locator	يشير إلى الموقع ويمكن الاتجاه في نظام الإحداثيات العالمي
Valuator	يوفر عوامل الضبط (التقيس)
Choice	الاختيار من خيرات مختلفة وإعادة عدد موجب
Stroke	يوفر مواقع متتالية في الإحداثيات العالمية
Pick	اختار كائن العرض، اختيار الحالة والمسار
String	يوفر رموز مرتبطة

2.5. تمارين

يعمل مستخدم في نظام إحداثياته العالمي الخاص والذي يمثل مساحة مربعة لديها أحديثات الأركان التالية [(80,50) , (50,-20)]. الصورة تتكون من مثلث لديه إحداثيات الرؤوس vertices التالية:{5,30-] و [50,30] و [10,-5-]. نقلت مساحة المستخدم إلى منفذ منظر مربع يمثل جزء من نافذة الشاشة وتستخدم إحداثيات الجهاز معرفة بوحدات عناصر الصورة. إحداثيات أركان الجهاز هي: ,(409,613)] [(613,818). كل الشاشة لديها عنوانين عناصر صورة من 0 إلى 1023 لكل من x و y.

أ. خطط بالتقريب ما سيظهر على الشاشة (تجاهل كل شي فد يوفره نظام التشغيل)

ب. ما هي عنصر الصورة المقابلة لنقطة تقاطع الإحداثيات [0,0] في نظام إحداثيات المستخدم؟

ج. النقل (التخطيط) بين نظام إحداثيات المستخدم ونظام إحداثيات الجهاز يعطى بالتالي

$$x' = Ax + B$$
$$y' = Cy + D$$

أوجد قيم A و B و C و D.

د. حرك المستخدم النافذة بمقدار 50 عنصر صورة في اتجاه y الموجب و بمقدار 100 عنصر صورة في اتجاه x الموجب. أوجد القيم الجديدة A و B و C و D. وأعطى مخطط لم سيظهر في النافذة على الشاشة.

ه. إن أرجع المستخدم النافذة لموقعها السابق ثم قام بعملية بتكبير "zoom in" بتغير نظام الإحداثيات العالمي، بحيث كبرت الصورة أربع مرات مقارنة بحجمها الأول (إي مرتين في كل اتجاه) مع بقاء نقطة تقاطع محاور الإحداثيات في مكانها. أعد حساب إحداثيات النافذة والثوابت أعلاه A و B و C و D.

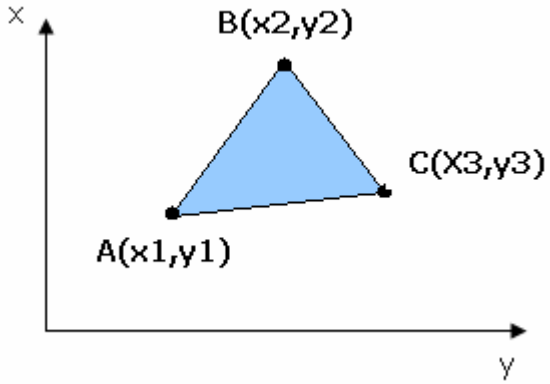
و. افترض إن كل يتخطى نافذة المستخدم يتم قصه من المنظر، أعطى مخطط لما سوف يظهر على الشاشة.

الوحدة الثالثة: المناظر الثنائية الأبعاد
Two Dimensional Views

3.1: تمثيل ألإشكال الثنائية الأبعاد
Representation of Two Dimensional Geometry

في التخطيط بالحاسوب يحدد شكل وحجم الكائن الثنائي الأبعاد باستخدام واصفات عددية ثنائية الأبعاد مرتبطة بنظام إحداثيات معين، عادة ما يكون نظام إحداثيات ديكارت Cartesian coordinate. يمكن استخدام عدد من التحويلات الهندسية لإزاحة وإعادة حجم ودوران الشكل أو الكائن.

العنصر الأساسي في أي نموذج ثنائي الأبعاد هي النقطة. فالخط مثلا يمثل بنقطتي النهاية والسطح يمثل بعدد من النقاط. أي تمثيل ثنائي الأبعاد يحدد بمجموعة من إحداثيات x و y، التي تعتبر عناصر النموذج. الشكل 3.1 يوضح تمثيل مثلث triangle باستخدام إحداثيات رؤوسه vertices.



الشكل 3.1: تمثيل المثلث باستخدام نظام إحداثيات xy

يمكن تمثيل هذا المثلث أيضا استخدام مصفوفة ترتيبها 3×2 كالاتى:

$$[P]_{TRIANGLE} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} \qquad 3.1$$

حيث أي جوز من xy تمثل متجه موقع position vector نسبة إلى نظام إحداثيات معين. إن استخدام المصفوفات مفيد في تحديد الشكل والتعامل معه في التخطيط بالحاسوب. إن بعض التحويلات تتم باستخدام تحقيقها باستخدام عمليات المصفوفات وبعض التحويلات الأخرى تتم بجمع المتجهات. يتطلب ذلك توحيد نظام الإحداثيات باستخدام نظام الإحداثيات المتجانس homogenous coordinateالذي نوقش في الوحدة السابقة.

3.2: تمثيل الخطوط والمنحنيات
أن الخطوة الابتدائية لرسم الخطوط والمنحنيات هي اختيار تمثيل مناسب لهم. توجد ثلاثة طرق لتمثيل الخطوط المنحنيات يمكن استخدامهم وهم

1. التمثيل باستخدام المعادلات الصريح Explicit: y=f(x)

وهنا يمثل الخط بالمعادلة

$$y = \frac{dy}{dx}(x - x_0) + y_0 \qquad 3.2$$

وتمثل الدائرة بالمعادلة

$$y = \pm\sqrt{r^2 - x^2} \qquad |x| \le r \qquad 3.3$$

2. التمثيل باستخدام المعادلات الوسيطة (البارامترية) x=(ft), y=f(t) : parametric equation
وهنا يمثل الخط بالمعادلة

$$x(t) = x_0 + t(x_1 - x_0)$$
$$y(t) = y_0 + t(y_1 - y_0)$$
$$t \in [0,1]$$

3.4

$$P(t) = P_0 + t(P_1 - P_0), \text{ or}$$
$$P(t) = (1-t))P_0 + tP_1$$

وتمثل الدائرة بالمعادلات التالية

$$x(\theta) = r\cos(\theta)$$
$$y(\theta) = r\sin(\theta) \qquad 3.3$$
$$\theta \in [0,2\pi]$$

3. التمثيل بالمعادلات الضمنية Implicit: f(x,y)=0
حيث يمثل الخط بالمعادلات التالية

$$F(x,y) = (x - x_0)dy - (y - y_0)dx \qquad 3.4$$

أذا كان F(x,y) =0 تكون (x,y) على الخط
F(x,y) >0 تكون (x,y) أعلى الخط
F(x,y) <0 تكون (x,y) أسفل الخط

ونمثل الدائرة باستخدام

$$x^2 + y^2 = r^2$$
$$F(x,y) = x^2 + y^2 - r^2 \qquad 3.5$$

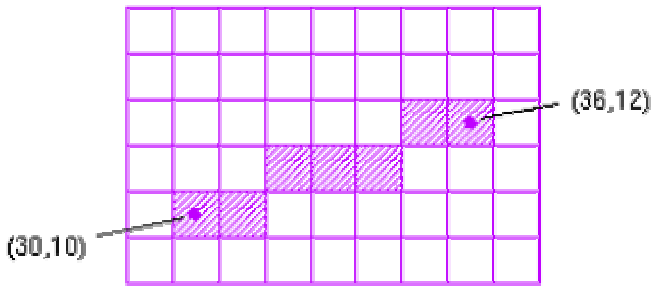
أذا كان F(x,y) =0 تكون (x,y) على االدائرة
F(x,y) 0 تكون (x,y) خارج الدائرة
F(x,y)<0 تكون (x,y) داخل الدائرة

3.3: الرسم الأمثل للخط

رسم الخط على شبكة مسح الخطوط raster grid يتطلب بعض التقديرات والمعالجة. تسمى هذه العملية بمسح الخطوط rasterization أو تحويل المسح scan-conversion. إن الخط المرسوم من عنصر الصورة (x1,y1) إلى عنصر الصورة (x2,y2) لابد إن تكون له الخصائص التالية (انظر الشكل 3.1):

- مستقيم straight
- يمر عبر النقاط النهائية
- سلس smooth
- لا يعتمد على ترتيب النقاط النهائية
- لديه سطوع منتظم

سطوع غير معتمد على الانحدار



الشكل 3.2: تخطيط الخط

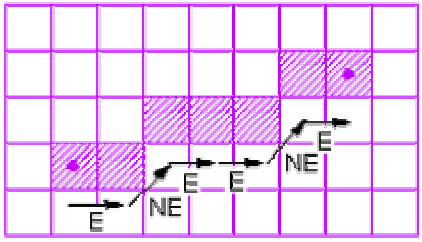
في المثال أدناه استخدمت خوارزمية مباشرة لرسم الخط المستقيم

```
DrawLine(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  float y;
  int x;

  for (x=x1; x<=x2; x++) {
    y = y1 +  (x-x1)*(y2-y1)/(x2-x1)
    SetPixel(x, Round(y) );
  }
}
```

يوضح المثال أدناه استخدام خوارزمية أفضل لرسم الخط المستقيم

```
DrawLine(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  float m,y;
  int dx,dy,x;

  dx = x2 – x1;
  dy = y2 – y1;
  m = dy/dx;
  y = y1 + 0.5;
  for (x=x1; x<=x2; x++) {
    SetPixel(x, Floor(y) );
    y = y + m;
  }
}
```

يمكن استخدام خوارزمية النقطة الوسيطة midpoint algorithm لرسم الخط المستقيم. إن خوارزمية النقطة الوسيطة أفضل من الخوارزميات المذكورة أعلاه والتي تستخدم حسابات الأعداد الصحيحة. لكل عنصر صورة يكون عنصر الصورة التالي E أو NE كما هو موضح بالشكل 3.2.



الشكل 3.2: خوارزمية النقطة الوسيطة

تستخدم خوارزمية النقطة الوسيطة هذا التعريف الغير مباشر للخط، F(x,y)=0. إن تحديد عنصر الصورة التالي E أو EN يتم كالآتي (انظر الشكل 3.3):
حدد

$$d = F(x_p + 1,\ y_p + 0.5$$

أذا كان d < 0 يكون الخط أسفل النقطة الوسيطة، اختار E
d > 0 يكون الخط أعلى النقطة الوسيطة، اختار NE
F(x,y)<0 يكون الخط على النقطة الوسيطة

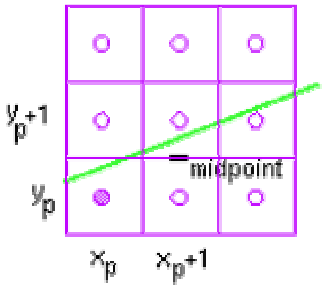
إن تم اختيار E

$$d_{new} = F(x_p + 2,\ y_p + 0.5)$$
$$d_{new} - d_{old} = F(x_p + 2,\ y_p + 0.5) - F(x_p + 1,\ y_p + 0.5)$$
$$\Delta d = d_{new} - d_{old} = dy$$

إن تم اختيار NE

$$d_{new} = F(x_p + 2,\ y_p + 1.5)$$
$$\Delta d = dy - dx$$

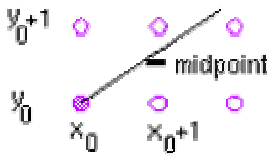
تكرر هذه العملية في خطوات على طول x لتحديد E أو NE



الشكل 3.3: تحديد عنصر الصورة التالي في خوارزمية النقطة الوسيطة

تحدد القيمة الابتدائية للخط d_{start} باستخدام المعادلة التالية (انظر الشكل 3.4)

$$\begin{aligned} d_{start} &= F(x_0 + 1,\ y_0 + 0.5) \\ &= (x_0 + 1 - x_0)dy - (y_0 + 0.5 - y_0)dy \\ &= dy - dx/2 \end{aligned}$$



الشكل 3.4: تحديد القيمة الابتدائية

لاستخدام خوارزمية أعداد صحيحة فقط، عرف

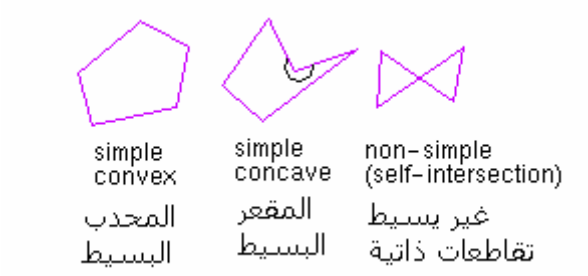
$$F'(x, y) = 2F(x, y)$$
$$d' = 2d$$
$$d'_{start} = 2dy - dx$$
$$\Delta d' = 2\Delta d$$

مثال:
الخوارزمية أدناه تنتج خط مستقيم يكون فيه x1 اقل من x2 والانحدار يساوى أو يقل عن 1:

```
drawline(x1, y1, x2, y2, colour)
int x1, y1, x2, y2, colour;
{
  int dx, dy, d, incE, incNE, x, y;

  dx = x2 - x1;
  dy = y2 - y1;
  d = 2*dy - dx;
  incE = 2*dy;
  incNE = 2*(dy - dx);
  y = y1;
  for (x=x1; x<=x2; x++) {
    setpixel(x, y, colour);
    if (d>0) {
      d = d + incNE;
      y = y + 1;
    } else {
      d = d + incE;
    }
  }
}
```

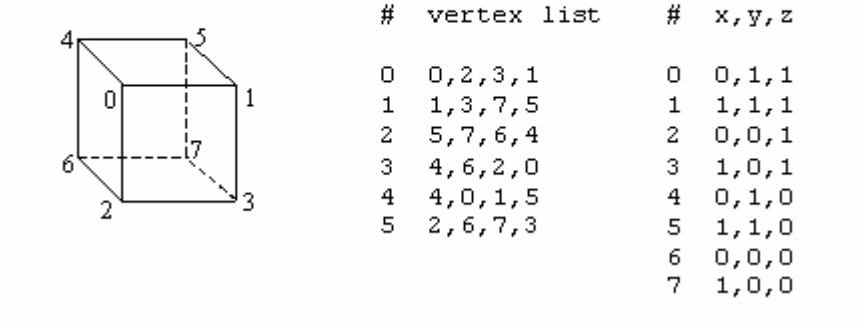
3.4. تمثيل المضلعات *Polygons*

يوجد أنواع مختلفة من المضلعات مثل (انظر الشكل 3.6)
المحدب البسيط simple convex
المقعر البسيط Simple concave
غير بسيط (التقاطع الذاتي) non-simple (self-intersection)



الشكل 3.5: أنواع المضلعات

من السهولة عمليا التعامل مع المثلث triangle حيث انه يكون دائما مستوى السطح ومحدب بسيط. يمكن استخدام العديد من الطرق لتمثيل المضلع، أكثر الطرق شيوعا واستخداما هي القائمة المراجع list of references التي ترتب قائمة الرؤوس vertex. إن استخدام هذه الطريقة يقلل من سعة التخزين والحسابات الزائدة عن الحاجة. كما يمكننا ربط وتوضيح كثير من المعلومات الأخرى مع كل رأس مثل المتعمدات normals والألوان colors وإحداثيات النسيجtexture coordinates. الشكل 3.6



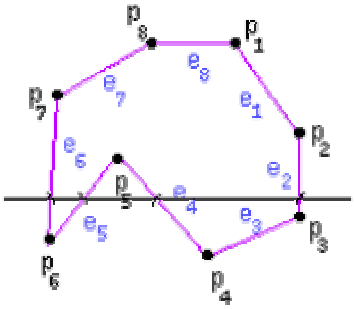
الشكل 3.6: طريقة قائمة المراجع

3.5. تحويل المسح *Scan Conversion*

المهمة الأساسية لتحويل المسح هي تظليل كل عناصر الصورة الموجودة داخل مضلع مقفول. يعتمد لون التعبئة في اغلب الأحيان على إضاءة lighting ونسيج texture ورؤية visibility المضلع الذي حول مسحه. هذه العوامل سيتم تجاهلها في الوقت الحالي وسنتعرض لها في الوحدات القادمة.

لنفترض إن المضلع مغلق ولديه حواف edges مرتبة، كما هو موضح بالشكل 3.7. إن خوارزمية تحويل المسح تكون على النحو التالي:

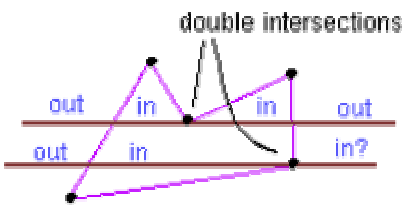
ادخل خط مع كل حافة
افرز التقاطعات مع x
احسب تكافؤ parity التقاطعات لتحديد الدخل والخرج
عبئ عناصر الصورة

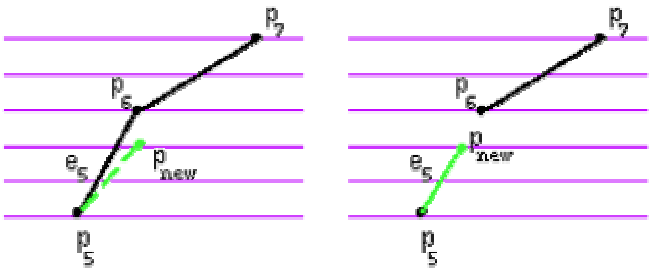


الشكل 3.7: تحويل مسح المضلع

توجد حالات خاصة لهذه الخوارزمية وهى:

الحافات الأفقية يمكن استثنائها
بالنسبة للرؤوس الواقعة على خط المسح
غير علامة y_i-y_{i+1}: تحسب مرتين (الشكل 3.7a)
لا تغير: قصر الحافة بخط مسح واحد (الشكل 3.7b)



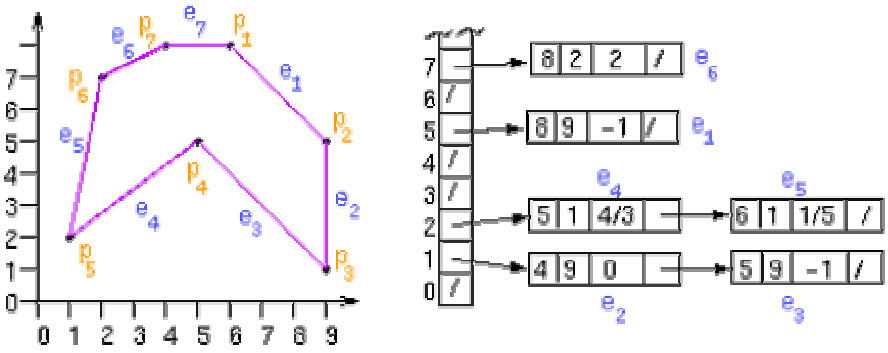


الشكل 3.6: الحالات الخاصة لخوارزمية مسح المضلع

كثير من اختبارات التقاطع يمكن التخلص منها بالاستفادة من الترابط المنطقي بين الخطوط الممسـوحة وهى
كل الحافات التي تقاطع الخط الممسوح y تقاطع أيضا الخط y+1
تغير قيمة x يمكن التنبؤ ب **الشكل 3.6: طريقة قائمة المراجع**
ها من خط المسح y إلى y+1

هياكل البيانات التالية يمكن استخدامها بواسطة خوارزمية تحويل مسح ذات كفاءة عالية:

جدول الحافة:
وتحدد هيكل البيانات هذا بالتالي (انظر الشكل 3.8)
- اعتبر حافات الانحراف traverse edge
- استثنى الحافات الأفقية
- إن لم يكن حافة قصوى، قصر الرأس الأعلى
- أضف الحافة إلى قائمة ربط خط المسح المقابل للرأس الأدنى، وفرز
  - قيمة y العليا: أخر خط مسح يأخذ في الاعتبار
  - قيمة x الدنية: بداية إحداثيات x للحافة
  - 1/m لزيادة قيمة x، تحسب قبل التقصير



الشكل 3.7: جداول الحافة

قائمة الحافة النشطة Active Edge List (AEL)
تمثل قائمة الحافة النشطة قائمة ربط للحواف النشطة في خط المسح الحالي y. أى حافة نشطة لديها المعلومات التالية:
- قيمة y العليا: أخر خط مسح يأخذ في الاعتبار
- قيمة x الدنية: بداية إحداثيات x للحافة
- 1/m لزيادة قيمة x، تحسب قبل التقصير

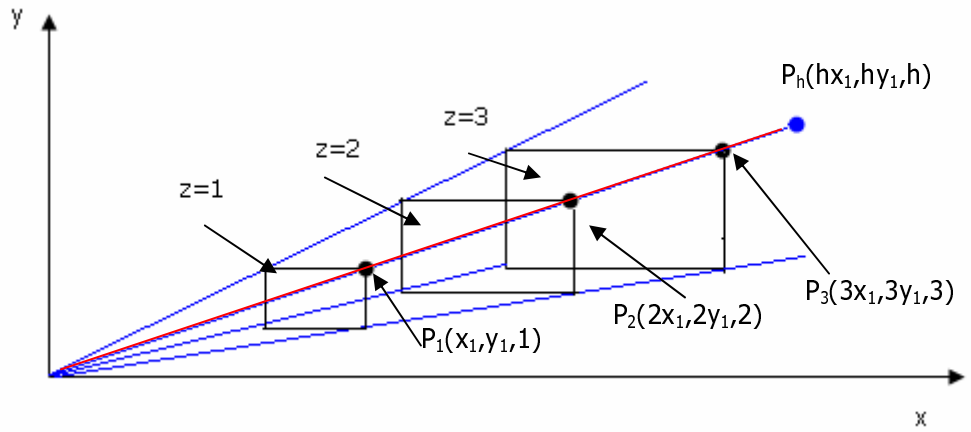
يتم فرز الحواف النشطة باستخدام x.

إن خوارزمية تحويل المسح تتكون من التالي:
- لكل خط مسح
- أضف الحافات في جدول الحافة المحدد في قائمة الحاقة النشطة

- إذا كانت قائمة الحافة النشطة غير فارغة
- صنف القائمة باستخدام x
- عبئ عناصر الصورة بين جوز الحافات
- امسح الحافات التي أنتهى التعامل معها
- حدث قيمة x لكل حافة

3.6. نظام الإحداثيات المتجانس *homogenous coordinate system*

حتى نتمكن من استخدام المصفوفات في كل عمليات التحويلات الهندسية نستخدم نظام الإحداثيات المتجانس homogenous coordinate system. إن استخدام نظام الإحداثيات المتجانس يعطى طريقة موحدة لوصف التحويلات الهندسية. لتوضيح مفهوم نظام الإحداثيات المتجانس، اعتبر نقطة النظام الثنائي الأبعاد $P_1(x_1,y_1)$ التي تقع في النظام الثلاثي ألأبعاد كما هو موضح في الشكل 5.3.



الشكل 3.8. التمثيل الثلاثي الأبعاد في الفضاء المتجانس

النقاط في الإشعاع الذي يصل النقطة P_1 إلى نقطة تقاطع محاور الإحداثيات يمكن التعبير عنه باستخدام العامل h على النحو التالي:

$$P(x,y,z) = P(hx, hy,h)$$

أي نقطة ثنائية الأبعاد ما عدا نقطة تقاطع محاور الإحداثيات (h=0) يمكن تمثيلها بأحد النقاط في الإشعاع في فضاء ثلاثي الأبعاد (يسمى الفضاء المتجانس homogenous space). بصورة عامة في نظام ألإحداثيات المتجانس، أي نقطة ديكارتية يمكن تمثيلها بهيئة زائدة باستخدام أربع إحداثيات (hx,hy,hz,h).

الإحداثيات العادية تناظر النقطة عند يعبر الإشعاع السطح z=1. عند استخدام نظام الإحداثيات المتجانس النقاط في المجال الثنائي الأبعاد تمثل بمصفوفة مرتبتها nx3، عندما تكون n عدد الرؤوس في الكائن. فمثلا المثلث يوصف بالمصفوفة التالية:

$$[P]_{TRIANGLE} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

3.7: التحويلات الثنائية الأبعاد

تشتمل التحويلات الهندسية حساب الإحداثيات الجديدة للنقاط التي تشكل الكائن من نقطة تقاطع محاور الإحداثيات إلى موقع التحويل.

3.7.1: ضبط الحجم scaling
يسمح تحويل إعادة الحجم بتغير الكائن سواء بالتمديد أو الانكماش. رياضيا يعبر عن تحويل إعادة للحجم للنقطة P(x,y) إلى النقطة P'(x',y') بالعلاقة التالية:

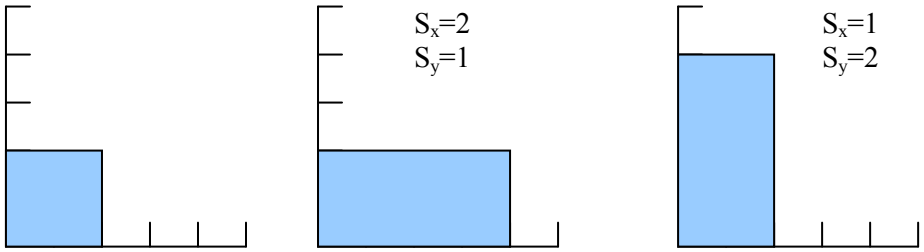
$x' = x \cdot S_x$ 3.6a

$y' = y \cdot S_y$ 3.6b

حيث s_x و s_y ثوابت تغير الحجم في اتجاه x و y على التوالي. يمكن كتابة هذه المعادلة على نسق المصفوفة كالآتي:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.7$$

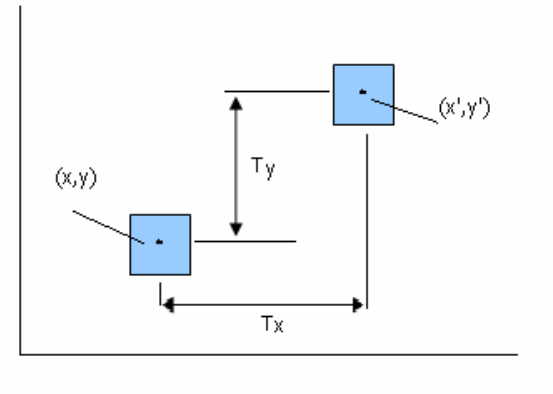
الشكل 3.9 يوضح أمثلة لتأثير تحويل الحجم على مربع. ويلاحظ عندما يكون عوامل إعادة الحجم متخلفة في اتجاه x و y يحدث تغير في حجم وشكل الكائن.



الشكل 3.9: تحويل ضبط الحجم

يسمى هذا التحويل ضبط الحجم حول نقطة تقاطع محاور الإحداثيات, ويسمى التحويل منتظم uniform إذا كان $S_x=S_y$.

3.7.2. تحويل الإزاحة Translation
تطبيق الإزاحة تغير موقع الكائن على العارض دون تغير حجمه أو شكله كما هو موضح بالشكل 3.10.



الشكل 3.10: تحويل الإزاحة

يعبر عن هذا التحويل رياضيا باستخدام

$x' = x \cdot T_x$ 3.8a

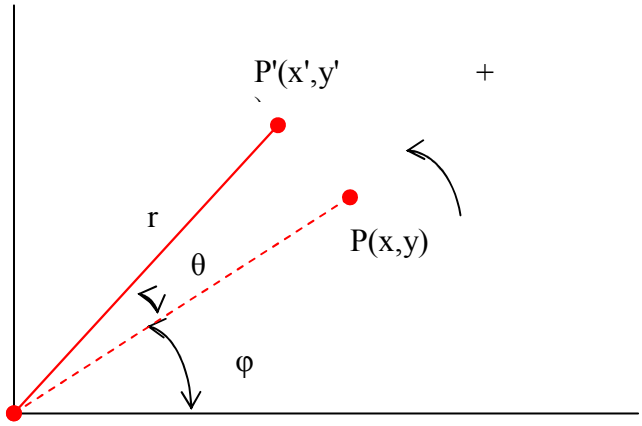
$y' = y \cdot T_y$ 3.8b

يمكن كتابة هذه المعادلة على نسق المصفوفة كالآتي:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} \qquad 3.9$$

3.7.3. تحويل الدوران Rotation

بتطبيق تحويل الدوران يتم لف الكائن بزاوية مقدارها θ حول نقطة تقاطع المحاور. الشكل 3.9 يوضح دورات النقطة P(x,y) بالزاوية إلى النقطة P'(x',y'). من المتعارف عليه إن كان الدوران في عكس اتجاه الساعة يكون الدوران موجب وإنا كان في اتجاه الساعة يكون سالب.



يمكن حساب موقع النقطة P'(x',y') باستخدام علاقات حساب المثلثات trigonometric التالية:

$$x = r\cos\varphi$$
$$y = r\sin\varphi \qquad 3.10$$

حيث العوامل r و φ موضحة في الشكل 3.9. وأيضا:

$$x' = r\cos(\varphi+\theta) = r\cos\varphi\cos\theta - r\sin\varphi\sin\theta$$
$$y' = r\sin(\varphi+\theta) = r\sin\varphi\cos\theta + r\cos\varphi\sin\theta \qquad 3.11$$

عوض المعادلة 3.10 في 3.11 نتحصل على

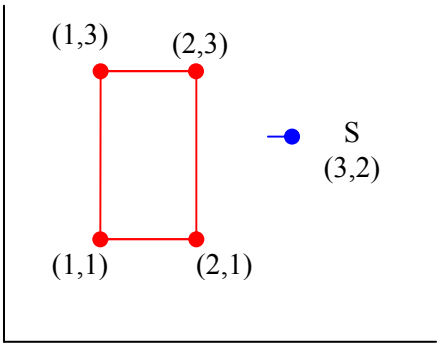
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta \qquad 3.12$$

يمكن كتابة هذه المعادلة على نسق المصفوفة كالآتي:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.13$$

من العادة قد يحتاج الشكل لكثر من تحول لمشاهدته، وفى مثل هذه الحالات يتم تركيب التحويل من التحويلات الموضحة أعلاه كما سيوضح في الأمثلة التالية.

مثال:
دور (لف) المستطيل المكون من النقاط $P_1(1,1)$ و $P_2(2,1)$ و $P_3(2,3)$ و $P_4(1,3)$ 30 درجة في عكس اتجاه الساعة حول النقطة S(3,2).

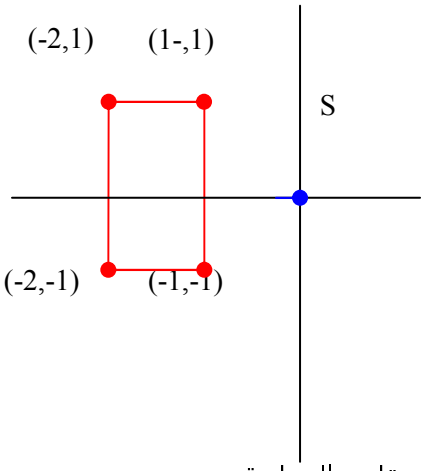


الحل:
يشتمل على الحل على التحويلات المتتالية التالية:

1. إزاحة النقطة S إلى نقطة تقاطع محاور الإحداثيات
   - مصفوفة التحويل

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & -2 & 1 \end{bmatrix}$$

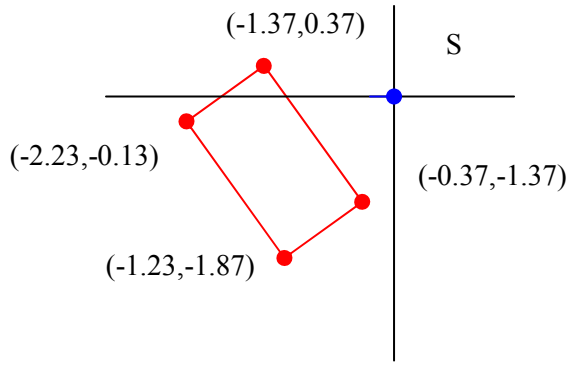
   - إزاحة النقطة S ينتج مباشرة إزاحة المستطيل لموقع جديد كما هو موضح أدناه



2. الدوران بثلاثين درجة عكس عقارب الساعة
   - مصفوفة التحويل

$$\begin{bmatrix} 0.8660 & 0.50 & 0 \\ -0.50 & 0.8660 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   - الشكل أدناه يوضح موقع المستطيل بعد الدوران

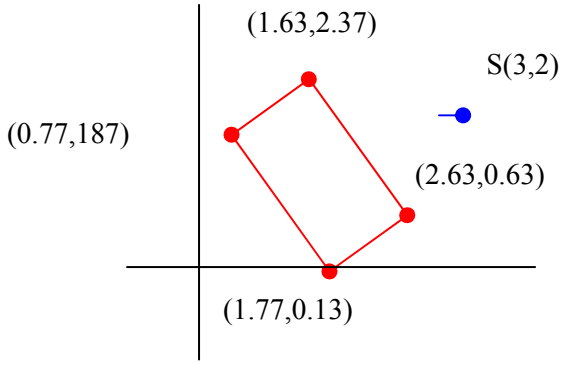


3. أراحة S مرة آخرة لموقعها ألأول

- مصفوفة التحويل

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

- الشكل أدناه يوضح الموقع الأخير المستطيل



مثال:
للنماذج التي توصف بعدد كبير من النقاط، يمكن استخدام طريقة حساب أكثر كفاءة للحصول على التحويل وذلك بضرب المصفوفات أولا ثم نستخدم مصفوفة النقطة. المثال أعلاه يحل كالاتى:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & -2 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 1.4 & -1.23 & 1 \end{bmatrix}$$

لإيجاد النقاط [P'] نستخدم

$$[P'] = [P][T] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 2 & 3 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 1.4 & -1.23 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.77 & 0.13 & 1 \\ 0.77 & 1.87 & 1 \\ 1.63 & 2.37 & 1 \\ 2.63 & 0.63 & 1 \end{bmatrix}$$

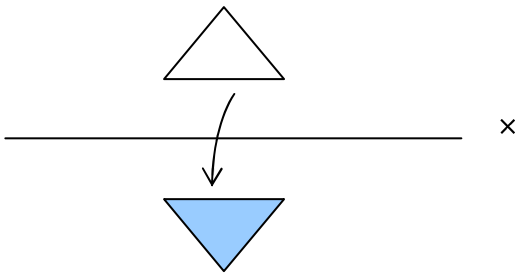
3.7.4. الانعكاس Reflection

يمكن فهم الانعكاس باعتبار الصورة في المرآة, ويستخدم تحويل الانعكاس لإنشاء الكمائن المتشابهة، حيث يمكن رسم نصف الشكل ثم استخدام الانعكاس لإتمام الشكل. قد يتم الانعكاس على نقطة أو خط. مصفوفة الانعكاس بالنسبة محور x أو y أو نقطة تقاطع المحاور يمكن كتابتها على النحو التالي:

$$[T_{REL}] = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.14$$

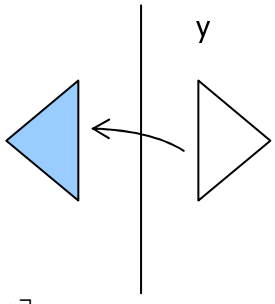
الأمثلة أدناه توضح حالات مختلفة لعكس مثلث

- على محور x: قيمة x لا تتغير وتنط قيمة y



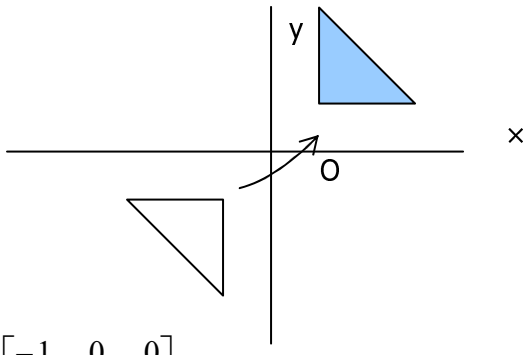
$$[T_{REL}]_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.15$$

- على محور y: قيمة y لا تتغير وتنط قيمة x



$$[T_{REL}]_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.16$$

- على نقطة تقاطع المحاور: تتغير قيمة كل م x و y



$$[T_{REL}]_O = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 3.17$$

3.7.5. تحويل القص Shearing

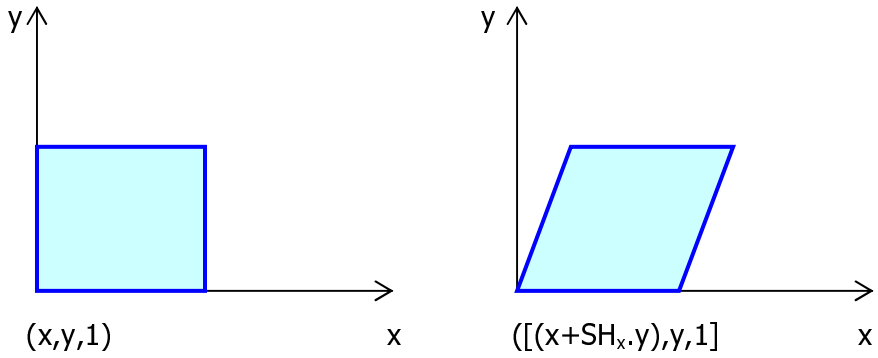
يغير تحويل القص قيمة الأحداث بإضافة دالة خطية للأحداث الأخر إليه. المصفوفة العامة للقص تكون على الهيئة التالية:

$$[T_{SH}] = \begin{bmatrix} 1 & b & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

سيوضح أدناه قص المربع لبعض الحالات.

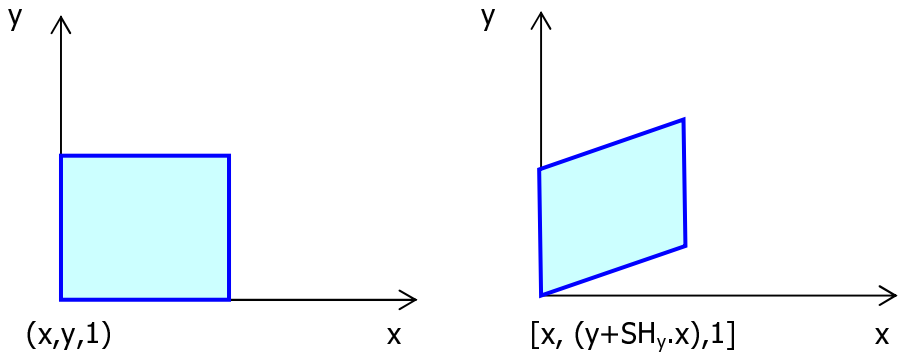
- قص اتجاه-x يؤثر فقط على إحداثيات x

$$[T_{SH}]_x = \begin{bmatrix} 1 & 1 & 0 \\ SH_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



- قص اتجاه-y يؤثر فقط على إحداثيات y

$$[T_{SH}]_y = \begin{bmatrix} 1 & SH_y & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



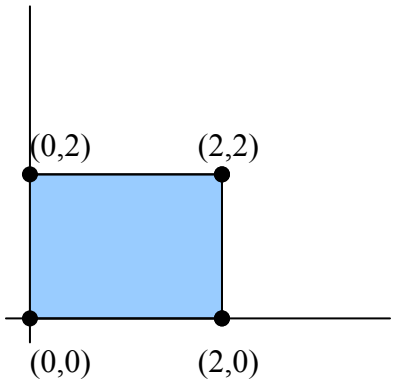
3.8: تمارين

1. معطى مثلث لديه إحداثيات الرؤوس التالية: A(3,1), B(1,3), C(3,3). لف (دور) المثلث بزاوية مقدارها 90^{o} على المحور الذي يمر عبر النقطة P(2,2) ثم حدد الإحداثيات الجديدة لكل من A و B و C.
2. وضح إن الدائرة التي تتعرض التحويل الذي يعطى بالمصفوفة

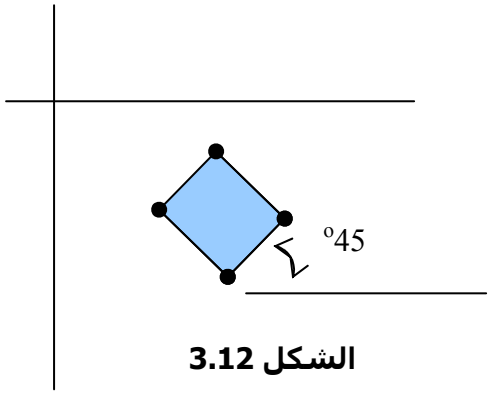
$$[T]=\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 0.75 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

لا تبقى دائرة بعد التحويل. ما هو الشكل بعد التحويل.

3. كبر المثلث المعطى في المساءلة الأولى إلى مرتين حجمه مع بقاء النقطة C في موقعها. أحسب إحداثيات النقاط الثلاثة بعد التحويل.
4. أشتق مصفوفة للانعكاس على الخط y=x.
5. اكتب متوالية من التحويلات على نسق مصفوفات التي تحتاجها لوضع المربع الموضح في الشكل 3.11 في الوضع الموضح في الشكل 12 مع تصغير حجمه للنصف. مركز المربع بعد التحويل عند النقطة (2,-2).



الشكل 3.11

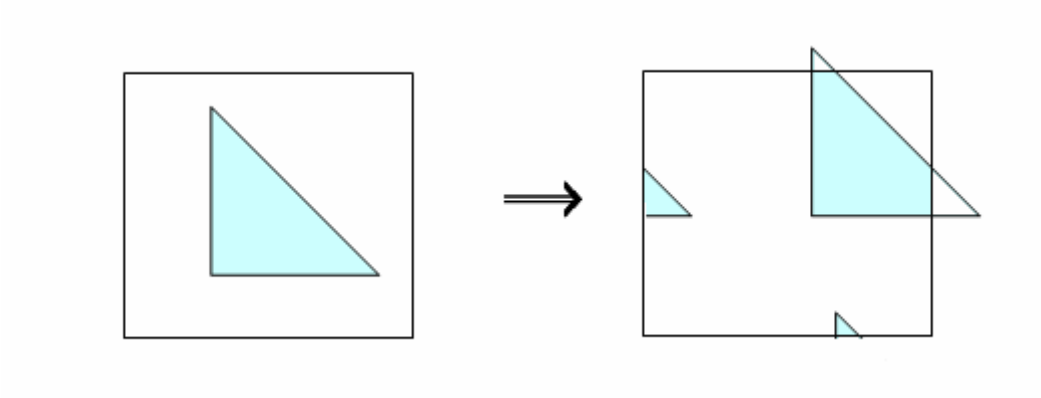


الشكل 3.12

الوحدة الرابعة: عمليات مشاهدة المناظر الثنائية الأبعاد
Two-Dimensional Viewing Operations

4.1. القطع (القص) الثنائي الأبعاد Two-Dimensional Clipping

عملية القطع تهتم بقص جزء من المنظر من النافذة، مثل قص صورة من مجلة. يمكن اعتبارها كتحويل هندسي مع قص أو قطع كل الخطوط والمنحنيات التي تعبر حدود النافذة. يقسم القص إي المنظر إلى جزئيين جزء مشاهد وآخر غير مشاهد. إن لم يتم القطع (أو القص) قد يظهر جزء المنظر الذي يقع خارج حدود (حافات) النافذة في الجهة المقابلة (العكسية) منها، كما هو موضح في الشكل 4.1. رغم حدوث هذا يقل مع استخدام تقنيات العتاد الحديثة فمن من الجيد التخلص من جزء المنظر الذي يقع خارج حدود النافذة (تأثير التغليف wraparound).



الشكل 4.1: تأثير التغليف Wraparound Effect

تعتمد خوارزميات القص بالكامل على شكل النافذة. في هذا المقرر ستفترض نافذة مستطيلة جنبتاها موازية لمحاور الإحداثيات الرئيسة.

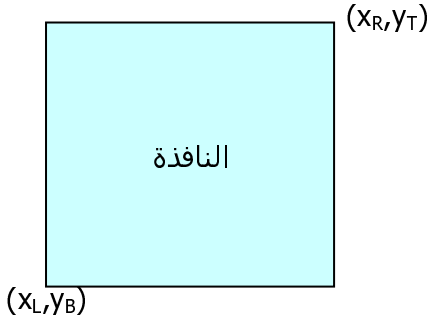
4.2. قص النقطة *Point Clipping*

يتحقق قص النقطة بسهولة بمضاهاة إحداثيات النقطة مع حدود النافذة. إذا كانت النافذة محددة بالأركان الأربعة يسار (L) ويمين ® وأعلى (T) وأسفل (B) كما هو موضح في الشكل 4.2، تكون النقطة مشاهدة إن استوفيت المتباينات inequalities التالية:

$$x_L \le x \le x_R \qquad 4.1a$$

$$y_B \le y \le y_T \qquad 4.1b$$

إن لم ستوفى إي من هذه المتباينات وسوف لا تعرض النقطة. رغم سهولة هذه الطريقة انه ليس من العملي تقسيم كل الشكل إلى نقاط. وعليه لابد من خوارزمية أكثر كفاءة تعتمد على الخطوط والمضلعات.



الشكل 4.2: حدود النافذة

4.3. قص الخط *Line Clipping*

1. الخوارزمية التي تستخدم لقص الخط تنقسم إلى جزئيين
   فحص كل مقاطع الخطوط وفصل المقاطع التي تتقاطع مع حدود النافذة

2. قطع (قص) المقاطع المتحصل عليها في الخطوة الأولى بحساب تقاطعها مع حدود النافذة

الخطوة الأولى يمكن تحقيقها بتقسيم الخطوط كالاتى:

- خطوط مشاهدة:

في هذه الحالة تكون نقاط نهاية المقطع بالكامل داخل حدود النافذة. مقطع الخط AB في الشكل 4.3 ضمن هذه المجموعة. ويتم عرض هذه الخطوط دون اى اختبارات أخرى.

- خطوط غير مشاهدة:

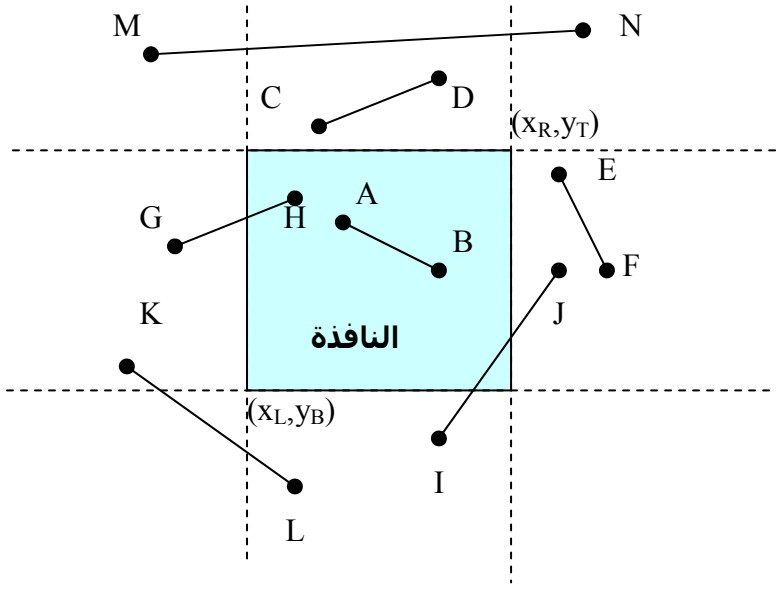
إذا كان نقاط النهاية الاثنين خارج حدود النافذة. يكون الخط من النقطة $P_1(x_1,y_1)$ إلى النقطة $P_2(x_2,y_2$ غير مشاهد إن لم يستوفى اى من المتباينات التالية:

$$\begin{aligned} &x_1 \text{ and } x_2 < x_L \\ &x_1 \text{ and } x_2 > x_R \\ &y_1 \text{ and } y_2 < y_B \\ &y_1 \text{ and } y_2 > y_T \end{aligned} \qquad 4.2$$

مقاطع الخطوط CD و EF تستوفى كل المتباينات في 4.2.

- غير محدد

لا يتضمن مقطع الخط في اى من الحالات أعلاه، ويجب أن تفحص للقص. المقاطع GH و IJ و KL في الشكل أمثلة لهذا النوع من الخطوط.



الشكل 4.3: مواقع مقاطع الخطوط المختلفة بالنسبة للنافذة

4.4. خوارزمية كوهين-ثازيرلاند Cohen-Sutherland

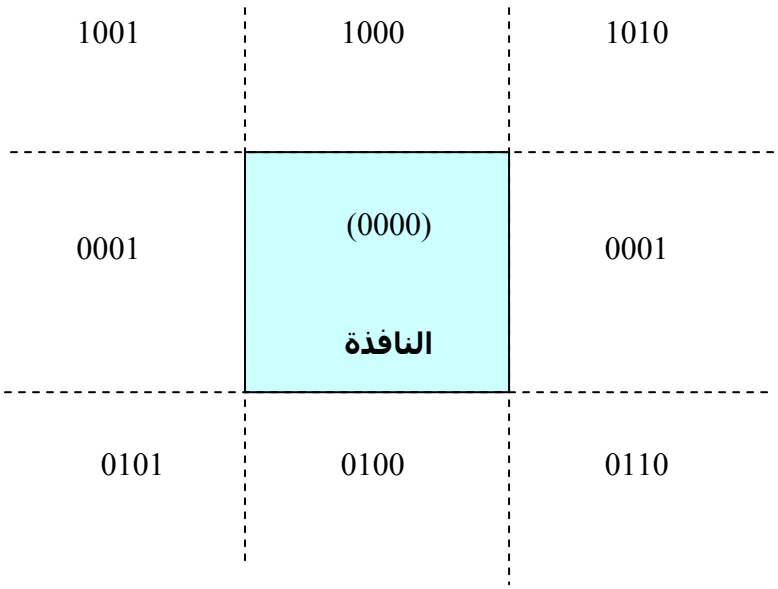
خوارزمية كوهين-ثازيرلاند تمثل إجراءات بسيطة وذات كفاءة عالية لتصنيف مقاطع الخطوط بالنسبة لحدود النافذة. تطبق الخوارزمية على مرحلتين:

المرحلة الأولى:

يحدد كل نقطة نهاية لمقطع الخط يراد فحصه شفرة code من 4 ثنائيات bits وذلك اعتمادا على المناطق التسع التي تحيط وتشتمل على النافذة، كما هو موضح في الشكل 4.4.

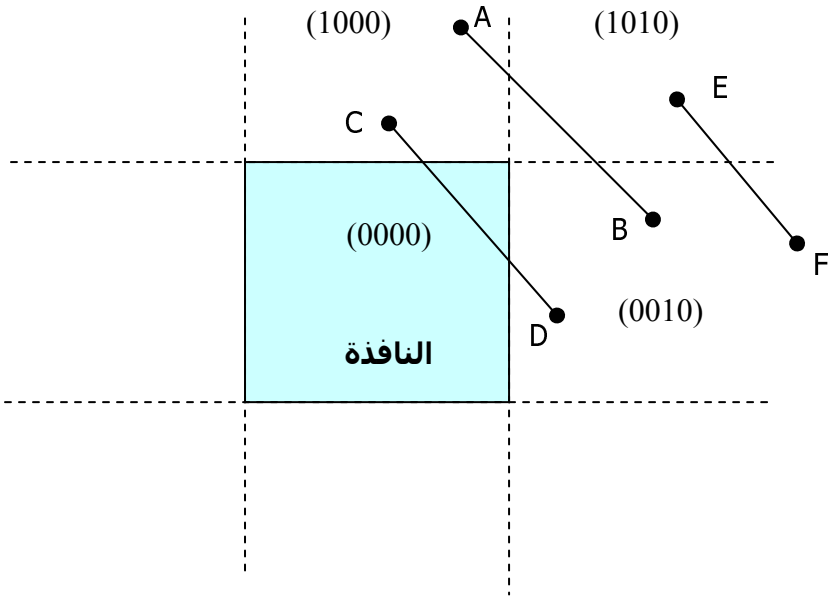
كل ثنائية في الشفرة تأخذ القيمة 1 (صحيح) أو القيمة 0 (خطاء) إبداء من أقصى اليسار على ضوء التالي:

- الثنائية ألأولى = 1 : نقطة نهاية المقطع أعلى النافذة
- الثنائية الثانية = 1 : نقطة نهاية المقطع أسفل النافذة
- الثنائية الثالثة = 1 : نقطة نهاية المقطع يمين النافذة
- الثنائية الرابعة = 1 : نقطة نهاية المقطع يسار النافذة



الشكل 4.4: شفرات مناطق النقطة النهائية

من البديهي إذا كان ثنائيات الشفرة تساوى (0000) تكون نقطة النهاية داخل النافذة أو حافتها. كل ثنائيات الشفرات تحدد بمقارنة إحداثيات نقطة نهاية مقطع الخط مع إحداثيات حدود (حافات) النافذة. الشكل 4.5 يعطى أمثلة لشفرة النقطة النهائية.



الشكل 4.5: أمثلة لشفرات النقاط النهائية

المرحلة الثانية:
يتم فحص النقاط النهائية كل مقاطع الخطوط مع بعضها البعض بناء على التالي:

- !ذا كانت نفس الثنائية تم ضبطها على 1 (صحيح) في كل من شفرات النقاط النهائية، يكون المقطع غير مشاهدة. فمثلا المقطع EF في الشكل 4.4 لديه شفرات نقاط نهائية تساوى (1010) و (0010) يقع على يمسن النافذة.
- أذا كانت ساوت الشفرة (0000) في كل من نقاط نهاية المقطع، يكون المقطع بالكامل داخل النافذة.
- إن كانت قيمة الثنائيات تساوى 1 في مواضع مختلقة من الشفرتين يكون المقطع غير محدد. مثل الخطوط AB و CD في الشكل 4.4 لديهم شفرات نقاط نهائية تساوى

(1000) و (0010). مثل هذه الخطوط قد لا تعبر النافذة، فمثلا الخط AB يكون غير مشاهد أما الخط CD فمشاهد جزئيا ويجب قصه.

لفحص هذه الخالة يستخدم منطق خاص يعرف بمنطق "ألإضافة المنطقية بطريقة الثنائية" bitwise logical AND، حيث يفترض إن التالي صحيح عند فحص شفرات النقاط النهائية ثنائية بثنائية:

نقطة النهاية 1 Endpoint 1		نقطة النهاية 2 Endpoint 2		ألإضافة المنطقية بطريقة الثنائية Bitwise Logical AND
0	and	0	← خطاء ←	0
0	and	1	← خطاء ←	0
1	and	0	← خطاء ←	0
1	and	1	← خطاء ←	1

بناء على هذا المنطق تحدد مشاهدة الخط على النحو التالي:

مشاهد ← تساوى كل من شفرات النقطتين النهائيتين (0000)

غير مشاهد ← تختلف الشفرات ولا تساوى ألإضافة المنطقية بطريقة الثنائية (0000)

غير محدد ← تساوى ألإضافة المنطقية بطريقة الثنائية (0000)، لكن لا تساوى كل شفرة نقطة نهائية على حده (0000)

باستخدام عملية الإضافة يتم فحص مقاطع الخطوط المعتبرة, الجدول 4.1 يعطى أمثلة للحسابات المطلوبة في هذه العملية.

جدول 4.1: حسابات المنطق

الخط	شفرات النقاط النهائية		الإضافة المنطقية	تعليق
1	1000	1001	1000	غير مشاهد
2	0101	0100	0100	غير مشاهد
3	0000	0000	0000	مشاهد
4	1001	0110	0000	فحص
5	0001	1000	0000	فحص

لفحص الخط نحتاج إلى تحديد تقاطعه مع حافة النافذة. ابسط طريقة لإيجاد التقاطع هي حل معادلات الخط والحافة. بالنسبة للنافذة المستطيلة والتي حافتاها موازية لمحاور الإحداثيات لا نحتاج لفحص الحافات الأربع في نفس الوقت. بتحديد ثنائية الشفرة التي لا تساوى 0 في ناتج ألإضافة المنطقية بطريقة الثنائية، تحدد حافة النافذة التي يتقاطع معها الخط كالاتى:

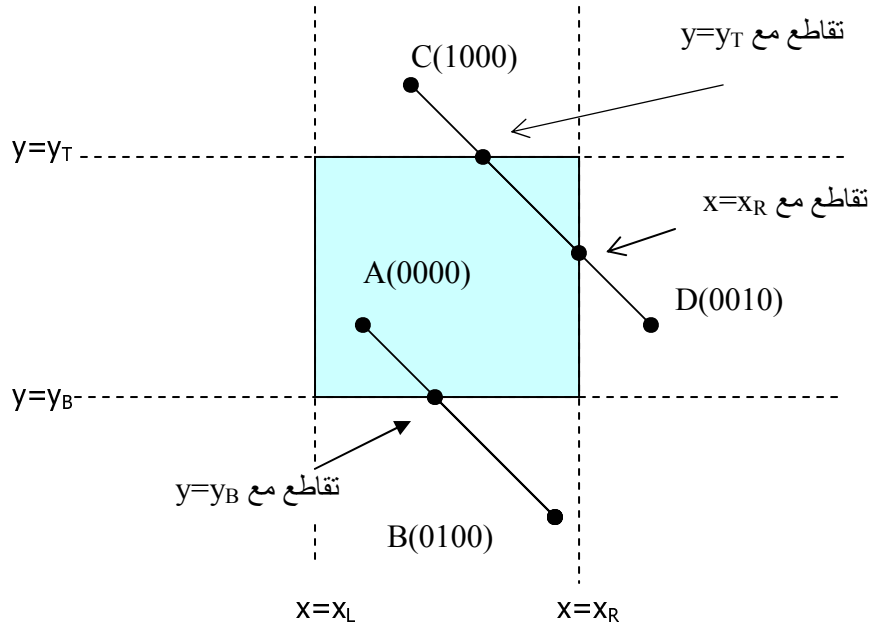
- الثنائية ألأولى = 1 : تقاطع مع $y=y_T$
- الثنائية الثانية = 1 : تقاطع مع $y=y_B$
- الثنائية الثالثة = 1 : تقاطع مع $x=x_R$
- الثنائية الرابعة = 1 : تقاطع مع $x=x_L$

الشكل 4.5 يوضح هذه العملية لخطيين. يوجد التقاطع بضبط المعادلات البارامترية لكل من مقطع الخط وحافة النافذة. معادلات الخط الذي يصل النقاط $P_1(x_1,y_1)$ و $P_2(x_2,y_2)$ هي:

$$x = x_1 + t(x_2 - x_1) \qquad 4.3a$$

$$y = y_1 + t(y_2 - y_1) \qquad 4.3b$$

عند حافة النافذة تصبح المعادلات x=constant و y=constant. وتحدد نقطة التقاطع بإيجاد قيمة t من احد المعادلات 4.3 والتعويض في الأخرى.



الشكل 4.5: حساب التقاطعات

مثال:

تعطى حافات نافذة مستطيلة بالتالي: $x_L=2$ و $y_B=2$ و $x_R=8$ و $y_T=8$. أفحص المشاهدة للمقاطع التالية باستخدام كوهين-ثيزرلاند وأن كان من الضروري قص مقاطع الخطوط عند حافة النافذة المناسبة.

الحل:

الخطوة الأولى: فحص الشفرات

- الخط AB
  - A(3,10) → (1000)
  - B(6,12) → (1000)
  - الإضافة المنطقية = (1000)
  - الخط غير مشاهد

- الخط CD
  - C(4,1) → (0100)
  - D(10,6) → (0010)
  - الإضافة المنطقية = (0000)
  - الخط غير محدد

الخطوة الثانية: قص الخط CD

- نقطة النهاية C لديها الشفرة (0100) حيث أن الثنائية الثانية لا تساوى صفر، يجب إيجاد التقاطع مع الحافة $y=y_B=2$. المعادلة البارمترية للخط CD

$$x = 4 + t(10-4) = 4 + 6t \qquad 1$$

$$y = 1 + t(6-1) \quad = 1 + 5t \qquad 2$$

عوض y=2 في المعادلة 2 نتحصل على قيمة t التالية:

$$t=1/5 = 0.2$$

وأيضاً

$$x = 4 + (1/5)(6) = 5.2$$

نقطة التقاطع هي $I_1(5.2,2)$

- نقطة النهاية D لديها الشفرة (0010) حيث أن الثنائية الثالثة لا تساوى صفر، يجب إيجاد التقاطع مع الحافة $x=x_R=8$. المعادلة البارمترية للخط CD

عوض x=8 في المعادلة 1 نتحصل على قيمة t التالية:

$$8 = 4+6t \rightarrow t=4/6 = 0.667$$

وأيضاً

$$y = 1 + (5)(2/3)= 4.33$$

نقطة التقاطع هي $I_2(8,4.33)$

حيث أن كل من I_1 و I_2 على حافات النافذة، تكون شفرات النهاية تساوى (0000) لكل منهم وعليه يكون مقطع الخط مشاهد بين نقاط التقاطع.

4.5. تمارين

1. لنفترض نافذة محددة بالإحداثيات التالية: (20,20-) و (60,60) في نظام الإحداثيات العالمي, اعتبر مقطع خط تعطى نقاط نهايته بالأتي: (30,0-) و (80,40).
   أ- اوجد الإضافة المنطقية لشفرات النقاط النهائية
   ب- اوجد تقاطع الخط مع حافة النافذة المناسبة
   ج- أفترض أن منفذ المشاهدة يحدد بالإحداثيات (10,30) و (200,130). اوجد إحداثيات منفذ المشاهدة لنقاط التقاطع.

2. في السؤال رقم 1، هل يحدث حجم النافذة مقارنة مع حجم منفذ المشاهدة تشويش في منظر الكائن؟ اشرح إجابتك.

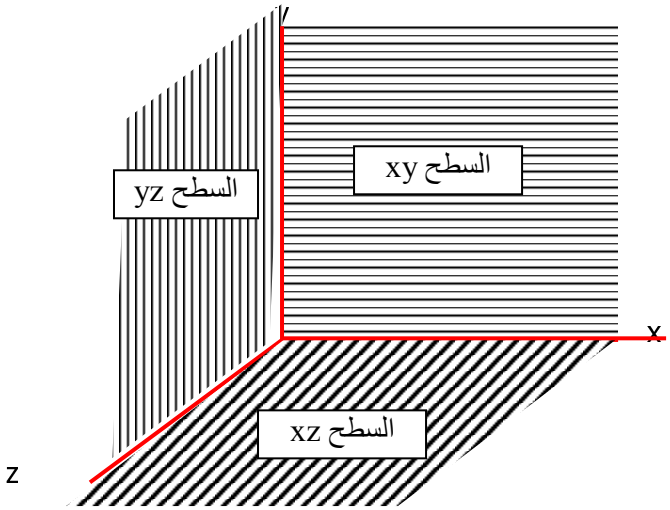
3. نافذة محددة بإحداثيات الركن الأسفل ألأيسر (2,2) وإحداثيات الركن الأعلى الأيمن (6,8), يراد قص مقطع الخط من A(3,1) إلى B(7,5) على هذه النافذة.
   أ- اوجد شفرات نقاط النهاية لمقطع الخط والإضافة المنطقية لهذه الشفرات
   ب- احسب نقاط تقاطع الخط مع حافات النافذة إن وجدت.

الوحدة الخامسة: التحويلات الهندسية الثلاثية الأبعاد
Three Dimensional Geometric Transformation

5.1. نظام الإحداثيات

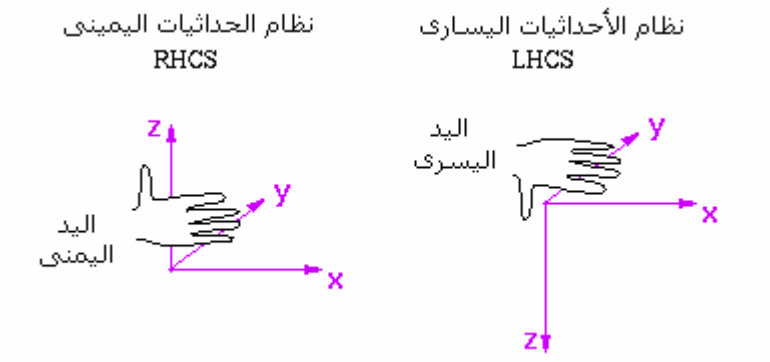
كما ذكر في الوحدة السابقة يمكن إنتاج نسخ مختلفة من المنظر باستخدام التحويلات. الكائن الثلاثي ألبعاد ينتج عادة في نظام إحداثيات عالمي ثلاثي الأبعاد ثم ينقل إلى العارض الذي يستخدم عادة نظام إحداثيات ثنائي الأبعاد. إن تغير ومشاهدة وإنتاج الكائن الثلاثي الأبعاد يحتاج إلى استخدام الهندسة الثلاثية الأبعاد وتحويلات الإحداثيات.

النقاط في الفضاء الثلاثي الأبعاد تمثل بنظام إحداثيات ثلاثي ألأبعاد والذي يمكن اعتباره امتدادا للنظام الثنائي الأبعاد الذي تم مناقشته في الوحدة السابقة. بإضافة محور-z متعامد مع السطح xy مما ينتج سطحيين أساسين أخريين هما xz و yz كما هو موضح في الشكل 5.1.



الشكل 5.1: تمثيل ثلاث أسطح أساسية متعامدة

توجه محاور الإحداثيات في النظام الثلاثي ألبعاد قد يكون يميني right-handed أو يساري left-handed، كما هو موضح بالشكل 5.2.



الشكل 5.2. توجه نظام الإحداثيات الثلاثي

5.2. الكمائن الثلاثية الأبعاد المحدودة بأسطح مستوية (الأوجه)
3-Dimensional Objects Bounded by Planar Surfaces (Facets)

تعرف الأوجه المستوية بترتيب مجموعة رؤوس ثلاثية الأبعاد على سطح واحد تكون مضلع مقفول closed polygon (ترسم خطوط مستقيمة من كل رأس إلى الرأس التالي مع ربط الرأس الأخير مع الأول). في الأشكال الثلاثية الأبعاد تتكون كل الأوجه من مضلعات مستوية planar polygons الشكل 5.3. البيانات التي تصف الوجه تتكون من نوعين:

- بيانات عددية numerical data وتعطى قائمة نقاط الأبعاد الثلاثية وعددها N×3 إن كان عدد النقاط N
- بيانات طوبولوجيا topological data وتحدد النقاط التي تربط مع بعضها لتكوين حافات الوجه

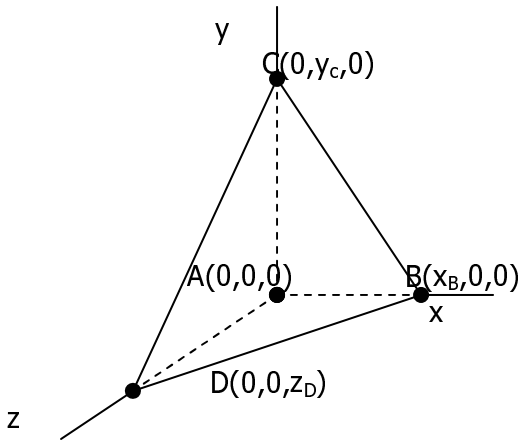
الشكل 5.3: شكل ثلاثي الأبعاد

إن استخدام الصيغ القانونية لكل من المسقط المنظور والمتعامد يسهل من عملية الحساب. يتطلب ذلك تحويل المشهد المحدد في إي نظام الإحداثيات الثلاثي بحيث يكون اتجاه المنظر على طول محور z (في حالة المسقط المنظور) ونقطة المنظر في تقاطع الإحداثيات. عامة نحتاج لتحويل إحداثيات إي نقطة في المشهد بحيث تكون نقطة منظر ما مختارة $\mathbf{C} = [C_x, C_y, C_z]$ في تقاطع محاور الإحداثيات. في كثير من الأحيان نحتاج لتحويل نقاط المشهد لأسباب أخرى مثل إنتاج تأثيرات خاصة في الصورة، مثل دوران الكائن. نتحصل على التحويلات من هذا النوع بضرب إي نقطة في المشهد بمصفوفة تحويل، بطريقة مماثلة لما تم مناقشته في الوحدة السابقة. وهناك مشكلة هنا، حيث انه لا يمكننا الحصول على تحويل عام باستخدام إحداثيات ديكارتي Cartesian coordinates . لهذا السبب نستخدم عادة نظام يعرف بالإحداثيات المتجانسة *homogeneous coordinates*، كما ذكر فى الوحدة السابقة.

تمثل النقاط في الفضاء الثلاثي الأبعاد باستخدام ثلاثة إحداثيات، الشكل 5.4 يوضح تمثيل الهرم المثلثى tetrahedron ، وفى نظام الإحداثيات المتجانس يكون للنقاط الثلاثية الأبعاد إحداث رأسي رابع كالاتى:

$$\mathbf{P} = [p_x, p_y, p_z, s]$$

الإحداث الرابع هو عامل تغير الحجم (تقيس) scale factor. إن تحويل الإحداثيات المتجانسة إلى الإحداثيات الديكارتية يتم بقسمها في إحداثيات رأسية أخرى، فالإحداثيات الديكارتية المكافئة إلى $[p_x, p_y, p_z, s]$ هي $[p_x/s, p_y/s, p_z/s]$. في أغلب الأحيان تكون قيمة العامل s تساوى 1. إن أهمية استخدام الإحداثيات المتجانسة كما ذكر في الوحدة السابقة، تتمثل في إنها تسمح لنا باستخدام إزاحة اى نقطة في المشهد باستخدام ضرب المصفوفات.



الشكل 5.4: تمثيل الهرم المثلثى tetrahedron

في نظام الإحداثيات المتجانسة يمثل الهرم المثلثى باستخدام مصفوفة ترتيبه 4×4 كالآتي:

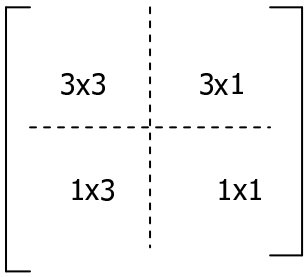
$$[\mathrm{P}] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ x_B & 0 & 0 & 1 \\ 0 & y_C & 0 & 1 \\ 0 & 0 & z_D & 1 \end{bmatrix} \qquad 5.1$$

5.3. التحويلات الهندسية الثلاثية ألإبعاد

تنفذ التحويلات في القضاء الثلاثي الأبعاد باستخدام نفس الأسلوب الذي استخدم تنفيذ التحويلات في المجال الثنائي الأبعاد مع إضافة إحداث-z. في نظام الإحداثيات المتجانس تمثل التحويلات في الفضاء الثلاثي ألأعاد باستخدام مصفوفة من النوع:

$$\begin{bmatrix} A & B & C & 0 \\ D & E & F & 1 \\ G & H & I & 0 \\ J & K & L & S \end{bmatrix} \qquad 5.2$$

يمكن تقسيم هذه المصفوفة كالاتى:



المصفوفة الفرعية التي ترتيبها 3×3 في اليسار الأعلى تسمح بضبط الحجم scaling والانعكاس reflection و القص shearing والدوران rotation. المصفوفة الفرعية بالترتيب 3×1 في اليسار الأدنى تنتج الإزاحة والمصفوفة الفرعية بالترتيب 1×1 في اليمين الأسفل تنتج ضبط حجم شامل منتظم "uniform global scalin". أما المصفوفة بالترتيب 3×1 في اعلي اليمين يمثل وجودها جزء من تمثل الإحداثيات المتجانسة.

5.4. ضبط الحجم scaling

النقطة P(x,y,z,1) يعاد ضبطها إلى النقطة P'(x',y',z',1) في نظام الإحداثيات المتجانسة بالتحويل التالي:

$$[\mathrm{x'} \quad \mathrm{y'} \quad \mathrm{z'} \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & E & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.3$$

يسمى هذا عادة إعادة الضبط بمرجعية نفطه تقاطع محاور الإحداثيات. إذا كانت العوامل A و E و I مختلفة يكون منظر الكائن مشوشا، إن تساوت هذه العوامل ينتج تغير الحجم. هذا النوع من ضبط الحجم الكلى يمكن أيضا أن ينتج من التحويل التالي:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix}$$
$$= [x \quad y \quad z \quad s] \qquad 5.4$$

لإيجاد الإحداثيات الديكارتية تعدل المعادلة 5.4 على النحو التالي:

$$[x \quad y \quad z \quad s] = \left[\frac{x}{s} \quad \frac{y}{s} \quad \frac{z}{s} \quad 1\right]$$

لقيمة s اكبر من واحد ينقص حجم المنظر. لتكبير المنظر العامل 1/s يجب إن يستخدم في مصفوفة ضبط الحجم. وتصبح الإحداثيات النهائية كالاتى:

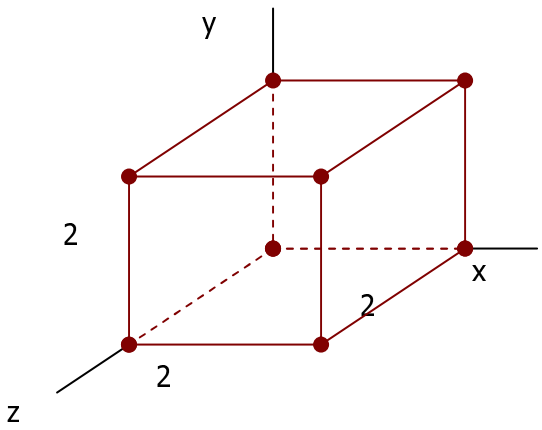
$$[sx \quad sy \quad sz \quad 1]$$

مثال:

اعتبر مكعب طول كل من أضلاعه 2 سم كما هو موضح في الشكل أدناه وتمثل الإحداثيات المتجانسة للمكعب بهيئة المصفوفة التالية:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \end{bmatrix}$$

استخدم مصفوفة ضبط الحجم لتصغير حجم المكعب لمكعب وحدة.



الحل:

مصفوفة ضبط الحجم المطلوبة لتحويل مكعب 2 سم لمكعب وحدة هي:

$$[T_{SC}]=\begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{a}$$

أو

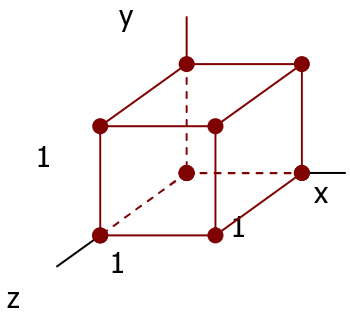
$$[T_{SC}]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad \text{b}$$

يمكن استخدام اى من المصفوفتان. باستخدام المصفوفة b نتحصل على:

$$[P']_{CUBE}^{UNIT} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 \\ 2 & 2 & 0 & 2 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \end{bmatrix}$$

في الهيئة المعيارية

$$[P']_{CUBE}^{UNIT} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$



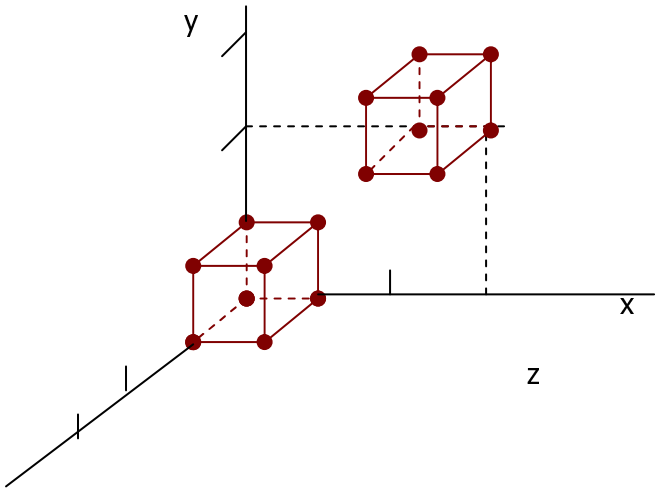
5.5. الإزاحة

مصفوفة التحويل التالية تحرك (تزيح) النقطة (x,y,z) إلى النقطة (x',y',z') عبر (J,K,L):

$$[\mathrm{x'} \quad \mathrm{y'} \quad \mathrm{z'} \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ J & K & L & 1 \end{bmatrix} \qquad 5.5$$

تمثل قيم كل من J و K و L الإزاحة النسبية للنقطة في اتجاهات x و y و ئ على التوالي.

الشكل 5.5 يوضح إزاحة مكعب وحدة لديه احد الرؤوس عند تقاطع محاور الإحداثيات لموقع جديد في الفضاء يحدد بمصفوفة الإزاحة.

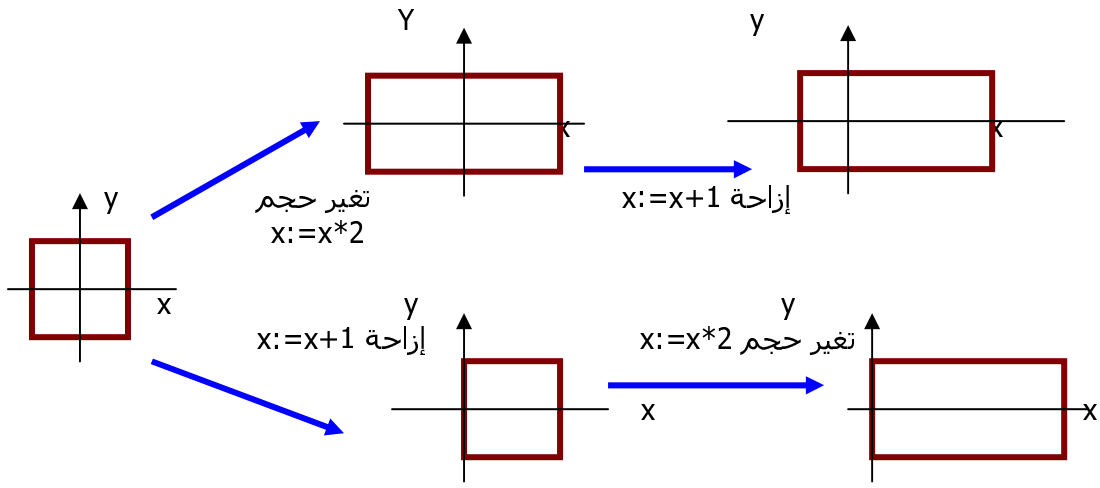


الشكل 5.5. إزاحة مكعب وحدة

أي رأس من رؤوس هذا المكعب يزاح بنفس القدر، عليه في هيئة المصفوفة تصبح إحداثيات المكعب المزاح

$$[P']_{CUBE}^{UNIT} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 2 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 1 \\ 3 & 2 & 0 & 1 \\ 3 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 \\ 3 & 3 & 0 & 1 \\ 2 & 3 & 0 & 1 \end{bmatrix}$$

ويلاحظ هنا أن تعرض الكائن إلى أكثر من تحويل مثل ضبط الحجم والإزاحة فمن الضروري القيام بهذه التحويلات بالترتيب الصحيح ولا يمكن عكسهما. يوضح الشكل 5.6 هذه الحالة باستخدام صورة بسيطة.



الشكل 5.6. أهمية ترتيب التحويلات

مثال:
أعتبر كائن عند نقطة تقاطع محاور الإحداثيات. كبر حجم الكائن إلى مرتين حجمه الأصلي وحركه إلى النقطة [5,5,20].

الحل:
ترتيب التحويل: تغير حجم ثم يتبع بإزاحة، يستخدم المصفوفات التالية

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 5 & 5 & 20 & 1 \end{bmatrix}$$

نضرب أولا مصفوفة الإزاحة لإزاحة النقاط

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 5 & 5 & 20 & 1 \end{bmatrix}$$

5.5. الدوران

الدوران في الأبعاد الثلاثة مهم لإعطاء صورة شاملة لمعرفة شكل الكائن والتحقق منه بالنظر من زواياه مختلفة. إن الدوران في ألأبعاد الثلاثة أكثر تعقيدا مقارنة بالدوران في المجال الثنائي الأبعاد نسبة لان الدوران حيث يجب تحديد محور للدوران بدلا عن نقطة. يتم التعامل مع الدوران بصورة مختلفة، حيث انه من الضروري تحديد محور الدوران. من الضروري الاهتمام باتجاه الدوران. المعادلات أعلاه تتبع عرف نظام محاور اليد اليسرى. مما يعنى أن محور y الموجب يكون رأسيا، ومحور x الموجب على اليمين و محور z الموجب إلى داخل الصفحة. في هذه الحالة يكون الدوران مع اتجاه دوران عقارب الساعة عندما ينظر إليه من ناحية الموجبة للمحاور والعكس صحيح بان الدوران يكون عكس دوران عقارب الساعة إن نظر إلية من الناحية السالبة للمحاور.

بتمديد المصفوفة التي استخدمت للدوران في المجال الثنائي الأبعاد، يمكن كتابة مصفوفة الدورات بزاوية θ حول محور-z عكس عقارب الساعة كالاتى:

$$[T_R]_z^\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.6$$

تنتج هذه المصفوفة الانتقال التالي

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned} \qquad 5.7$$

ان كان الدوران مع عقارب الساعة تصبح مصفوفة الدوران حول المحور-z

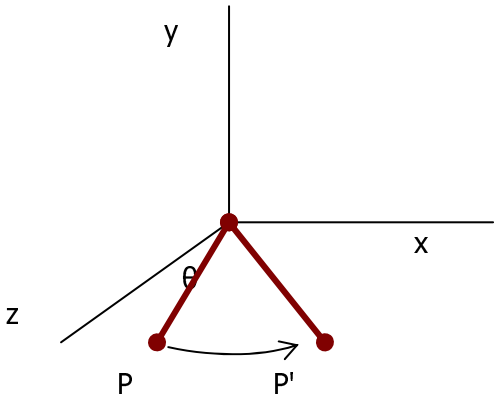
$$[T_R]_z^\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.8$$

بطريقة مماثلة يمكن إيجاد الدوران بزاوية θ حول محور-y عكس عقارب الساعة، باعتبار الشكل 5.7

$$[T_R]_y^\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.9$$

تنتج هذه المصفوفة الانتقال التالي

$$\begin{aligned} x' &= x\cos\theta + z\sin\theta \\ y' &= y \\ z' &= -x\sin\theta + z\cos\theta \end{aligned} \qquad 5.10$$

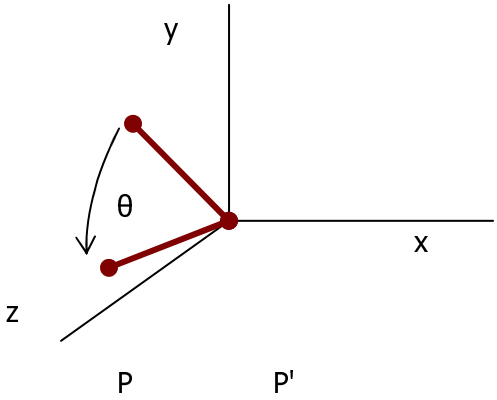


الشكل 5.7. الدوران حول محور-y

إن كان الدوران مع عقارب الساعة تصبح مصفوفة الدوران حول المحور-y

$$[T_R]_y^\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.11$$

يمكن اشتقاق مصفوفة الدوران حول المحور-x عكس عقارب الساعة من الشكل 5.8



الشكل 5.8. الدوران حول محور-x

$$[T_R]_x^\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.12$$

تنتج هذه المصفوفة الانتقال التالي

$$\begin{aligned} x' &= x \\ y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \end{aligned} \qquad 5.13$$

إن كان الدوران مع عقارب الساعة تصبح مصفوفة الدوران حول المحور-x

$$[T_R]_x^\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5.14$$

مقلوبات inverse هذه المصفوفات يمكن أن يتم من غير استخدام طريقة حذف غاوسين Gaussean elimination وذلك باعتبار معنى إي تحويل. لمصفوفة تغير الحجم نستبدل القيم s_x و s_y و s_z بالقيم $1/s_x$ و $1/s_y$ و $1/s_z$ على التوالي. ولمصفوفة الإزاحة نستبدل القيم t_x و t_y و t_z بالقيم $-t_x$ و $-t_y$ و $-t_z$ على التوالي. لمصفوفة الدوران يلاحظ التالي:

$$\cos(-\theta) = \cos(\theta) \text{ and } \sin(-\theta) = -\sin(\theta)$$

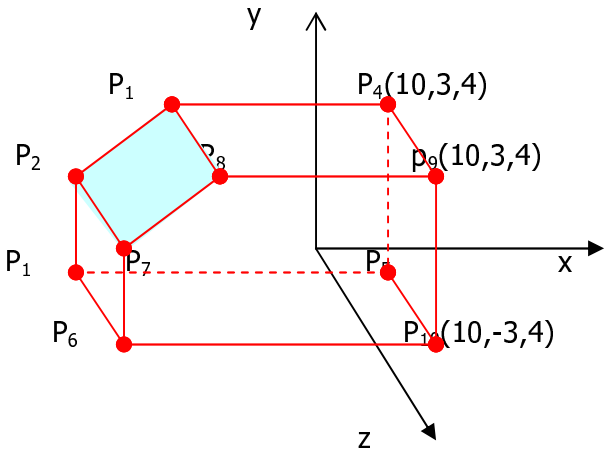
وعليه نتحصل على مقلوب المصفوفة بتغير إشارة حدود جيب الزاوية sin terms.

مثال:

أعتبر القالب المشطوف الموضح في الشكل 5.8. النقاط P_1 و P_2 تحدد شكل (هندسة) القالب ويمكن كتابتها في هيئة المصفوفة التالية:

$$[P]=\begin{bmatrix} -10 & -3 & -4 & 1 \\ -10 & 1 & -4 & 1 \\ -8.5 & 3 & -4 & 1 \\ 10 & 3 & -4 & 1 \\ 10 & -3 & -4 & 1 \\ -10 & -3 & 4 & 1 \\ -10 & 1 & 4 & 1 \\ -8.5 & 3 & 4 & 1 \\ 10 & 3 & 4 & 1 \\ 10 & -3 & 4 & 1 \end{bmatrix}$$

دور (لف) القالب بحيث يكون السطح الأسفل المحدد بالنقاط P_1 و P_5 و P_{10} و P_6 موازى للسطح xy.



الحل:
لجعل السطح المحدد بالنقاط P_1 و P_5 و P_{10} و P_6 موازى للسطح xy، يتطلب الدوران 90^0 عكس عقارب الساعة حول المحور-x، المصفوفة المناسبة تعطى بالأتي:

$$[T_R]_x^{\theta}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

باستخدام $\theta = 90^0$ تصبح المصفوفة

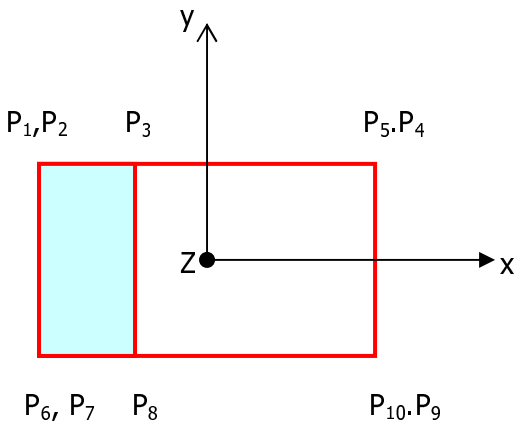
$$[T_R]_x^{90^o}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

مصفوفة النقاط المحولة تعطى بالتالي:

$$[P] = [P][T_R]_x^{90^o}$$

$$= \begin{bmatrix} -10 & -3 & -4 & 1 \\ -10 & 1 & -4 & 1 \\ -8.5 & 3 & -4 & 1 \\ 10 & 3 & -4 & 1 \\ 10 & -3 & -4 & 1 \\ -10 & -3 & 4 & 1 \\ -10 & 1 & 4 & 1 \\ -8.5 & 3 & 4 & 1 \\ 10 & 3 & 4 & 1 \\ 10 & -3 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -10 & 4 & -3 & 1 \\ -10 & 4 & 1 & 1 \\ -8.5 & 4 & 3 & 1 \\ 10 & 3 & -4 & 1 \\ 10 & 4 & -3 & 1 \\ -10 & -4 & -3 & 1 \\ -10 & -4 & 1 & 1 \\ -8.5 & -4 & 3 & 1 \\ 10 & -4 & 3 & 1 \\ 10 & -4 & -3 & 1 \end{bmatrix}$$

الشكل أدناه يوضح منظر الكائن بعد الدوران 90^0 عكس عقارب الساعة حول محور-x كما يشاهد على محور-z.

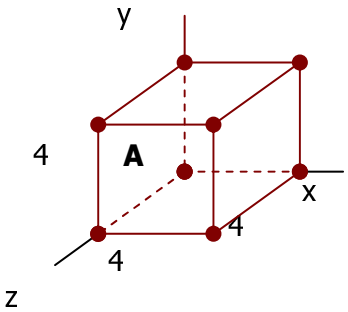


5.6. تمارين:

1. لف الخط المستقيم الذي يصل النقاط A(1,1,1) و B(4,3,2) 30^0 حول محور-x و 60^0 حول محور-y في الحالتين الدوران عكس عقارب الساعة. اعد العملية في الترتيب العكسي اى الدوران حول محور-y أولا ثم الدوران حول محور-x، قارن الإحداثيات النهائية في الحالتين.

2. إعتبر المكعب الموضح في الشكل أدناه مع طول الضلع يساوى 4 سم.
   أ. انقص حجم المكعب إلى النصف (طول الضلع يصبح 2 سم)
   ب. أزح المكعب المتحصل عليه لنكون النقطة A عند (1,1,1)
   ج. دور المكعب الموضح بزاوية 45^0 عكس اتجاه عقارب الساعة حول محور-z،

   ارسم منظر المكعب المتحصل عليه في كل حالة.

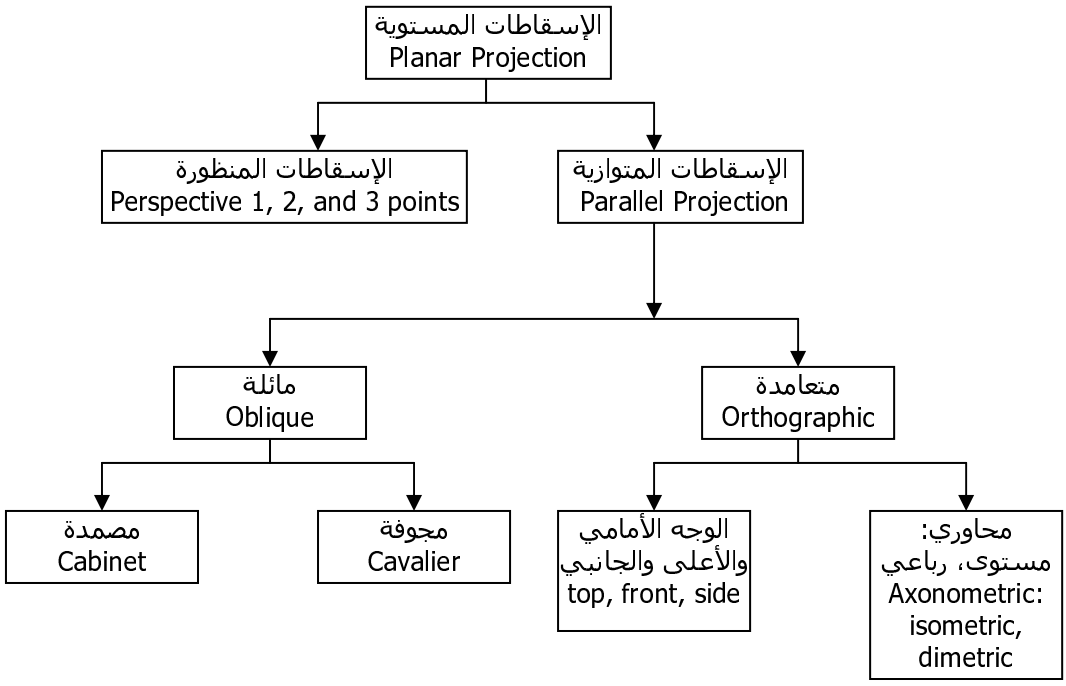


الوحدة السادسة: ألإسقاط والقص
Projection and Clipping

6.1. إسقاطات نماذج الإطار السلكي
Projections of Wire-Frame Models

يعرف الوضع (التخطيط) Mapping: $f: R^n \rightarrow R^m$. الإسقاط projection حالة خاصة من التخطيط حيث تكون m اقل من n. ويستخدم الإسقاط السطحي planar Projection لوضع الأشكال الثلاثية الأبعاد على سطح.

توجد أنواع مختلفة للإسقاط المستوية، كما هو موضح بالشكل 6.1



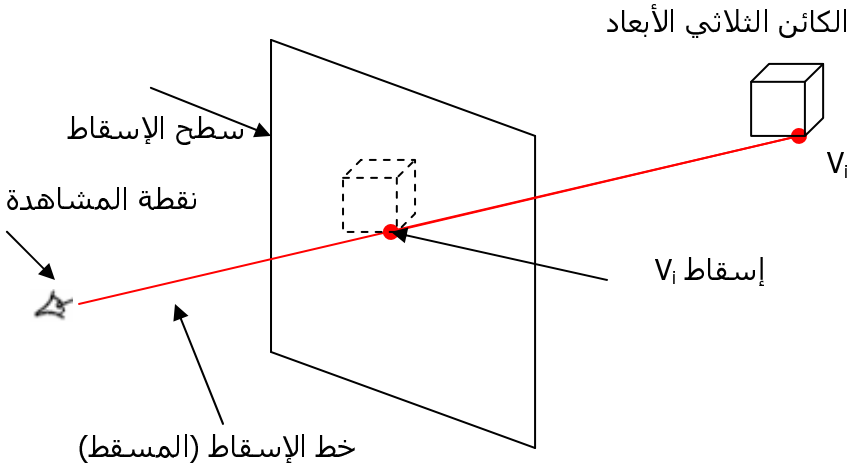
الشكل 6.1: أنواع الإسقاطات المستوية

جهاز العرض في نظم التخطيط الحاسوبية ثنائي الأبعاد ويتطلب منا هذا تعريف تحويل من الفضاء الثلاثي الأبعاد إلى السطح جهاز العرض الثنائي الأبعاد. هذا التحويل يعرف بالإسقاط projection. ويتم مشاهدة الكائن الثلاثي الأبعاد بالتحويل من الفضاء الثلاثي الأبعاد إلى السطح الثنائي الأبعاد ثم تستخدم بدائيات primitives الرسم البسيطة لرسم الكائن.

لإسقاط كائن الثلاثي الأبعاد على سطح ثنائي الأبعاد يتطلب اختيار سطح الإسقاط projection surface أولا وثم تحديد المساقط projectors أو الخطوط التي تمر عبر كل رأس للكائن.

تضع رؤوس الإسقاط projection vertices عند تقاطع المسقط مع سطح الإسقاط. إن ابسط وأكثر طرق الإسقاط استخداما لمشاهدة للأشكال الثلاثية الأبعاد هي استخدام أسطح أسقاط مستوية وخطوط مستقيمة للمساقط. ويسمى هذا بالإسقاط الهندسي المستوى planar geometric projections، الشكل 6.2. ينشا اتجاه المشاهدة من المشاهد إلى الكائن بواسطة خطوط ساقطة (مساقط) تمر عبر السطح (سطح الإسقاط) الذي سيظهر عليه الساقط.

عموما يمكن الإسقاط على أسطح مختلفة مثل السطح الكروي والقمع الخ، كما يمكن استخدام المساقط المنحنية لإنتاج التأثير العدسة. وسنتعرض في هذا المقرر فقط للإسقاط المستوى الخطى الذي يمثل ابسط نسق للنظر للكائن حيث ترسم كل الحافات المسقطة.



الشكل 6.2: الإسقاط المستوى

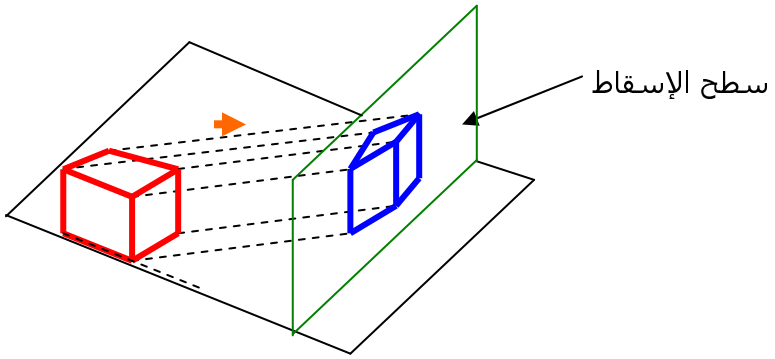
يوجد نوعين من الإسقاطات الهندسية المستوية:

- الإسقاطات المتوازية parallel projection
- الإسقاط المظورى perspective projection

6.2. ألإسقاطات المتوازية *Parallel Projection*

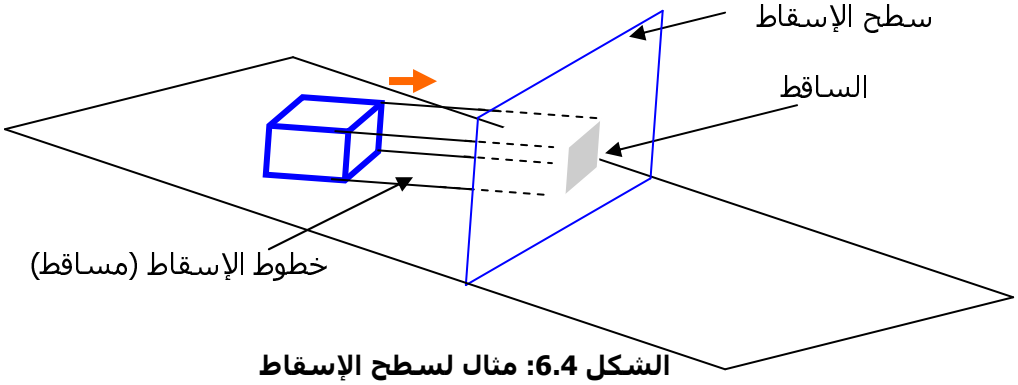
تستخدم هنا مساقط متوازية و يكون مركز الإسقاط centre of projection في اللاتناهى infinity وعليه تصبح المساقط موازية لبعضها البعض كما ذكر أعلاه. يوجد نوعين من المساقط يمكن استخدمهما في الإسقاطات المستوية وهم

- المساقط مائلة oblique بالنسبة لسطح الإسقاط،وهنا يجعل احد أسطح الكائن موازى لسطح الإسقاط، كما هو موضح بالشكل 6.3.



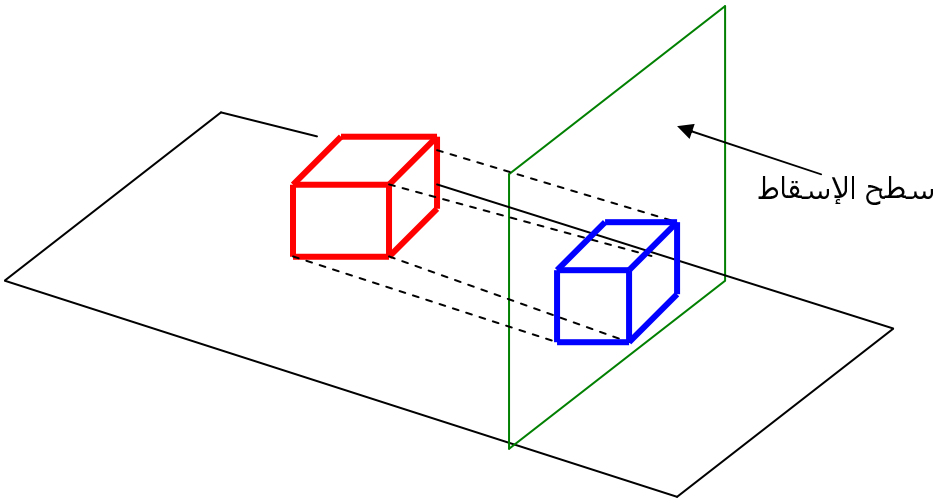
الشكل 6. 3: مثال للمساقط المائلة

- المساقط متعامدة orthographic، حيث تكون المساقط متعامدة مع سطح الإسقاط والذي قد يكون
  - متعامد متعدد المشاهد multivew orthographic، حيث تبقى أسطح الكائن موازية لسطوح الإسقاط الرئيسة، كما هو موضح في الشكل 6.4،



الشكل 6.4: مثال لسطح الإسقاط

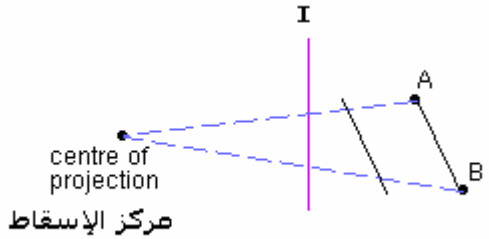
- إسقاط محاوري axonometric، يكون سطح الكائن مائلا بالنسبة لسطح الإسقاط، كما هو موضح الشكل 6.5.



الشكل 6.5: الإسقاط المحاورى

6.3. الإسقاطات المنظورة *perspective projections*

في الإسقاط المنظورى يقع مركز الإسقاط على مسافة محددة من سطح الإسقاط وتمر كل خطوط الإسقاط (المساقط) عبر هذه النقطة والتي تدعى نقطة المنظر viewpoint، كما هو موضح بالشكل 6.6. الإسقاط المنظورى يرسم مقصرا الخطوط foreshortening وهو غير خطى non-linear

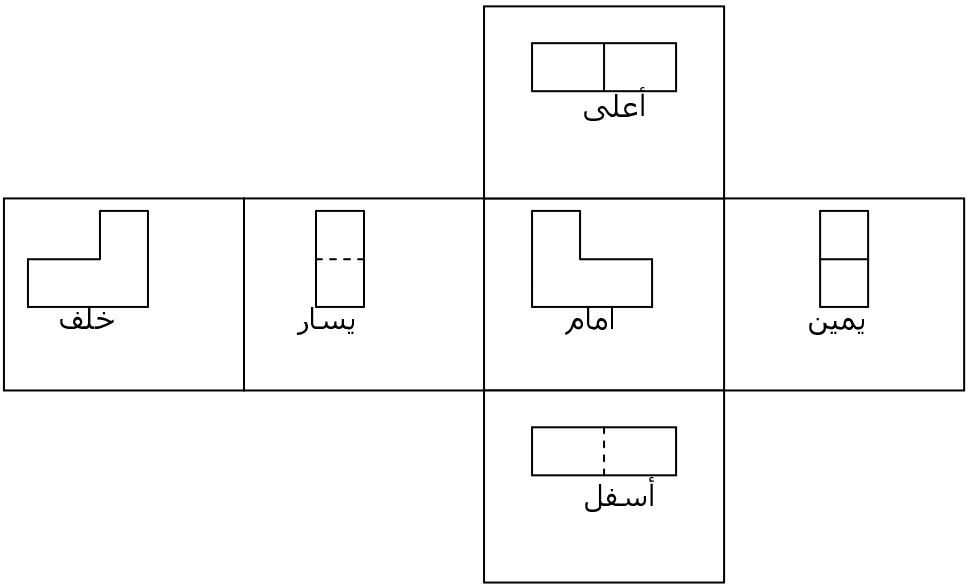


الشكل 6.6: الإسقاط المنظور

لتقليل الخلط الذي قد يحث عند التعامل مع مسائل ألإسقاط، يجب علينا معايرة أسطح الإسقاط بجعلها دائما موازية للسطح z=0 (السطح الذي يحتوى على المحور x والمحور y). إن هذه المعيارية لا تحد من العمومية، فإذا كان سطح الإسقاط غير موازى للسطح z=0 يمكننا استخدام تحويلات الإحداثيات في الفضاء الثلاثي الأبعاد لجعله موازى للسطح z=0. سيشاهد ي هذا المقرر منظر الكائن في النصف الموجب من الفضاء (z > 0) فقط، وعليه إن الإسقاطات التي تبدءا في الرؤوس ستمتد في اتجاه z السالب.

6.5. الإسقاط المتعامد المتعدد المشاهد

يعتبر الإسقاط المتعامد أكثر أنواع الإسقاطات استخداماً في التخطيط بالحاسوب وخاصة في مجال العلوم الهندسية وذلك لإعطاء مشاهد متعددة للكائن، كما هو موضح فى الشكل 6.7.



الشكل 6.6: تمثيل الإسقاط المتعامد المتعدد المشاهد

في الشكل 6.8، اعتبر النقطة P(x,y,z) والتي أسقطت على السطح-xy . نقطة الإسقاط P هي النقطة P'(x',y',0). التحويل الذي ينتج هذا الإسقاط يمكن التعبير عنه في هيئة المصفوفة على النحو التالي:

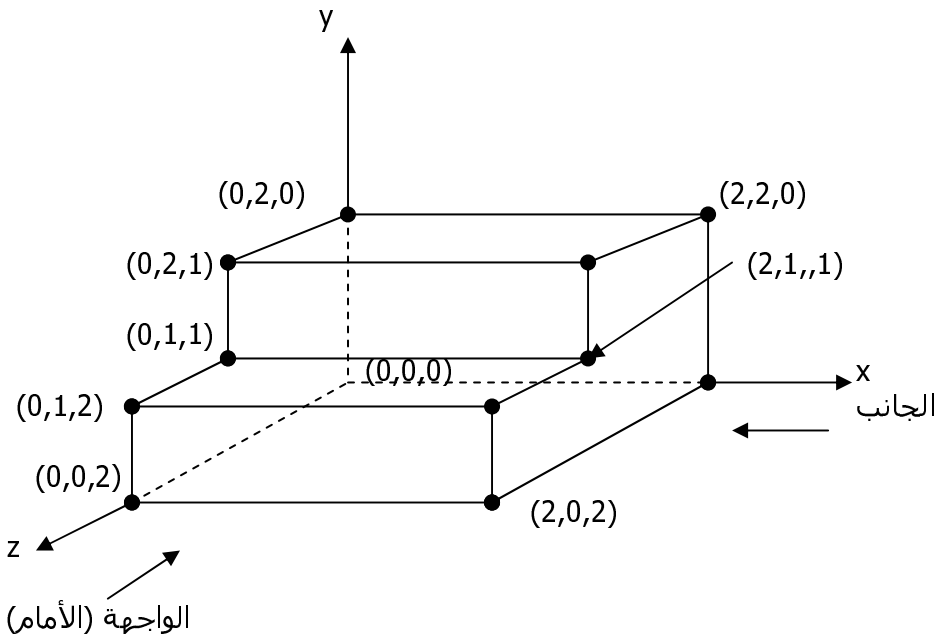
$$\begin{bmatrix} x' & y' & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 6.1$$

نفس الأسلوب يمكن أن يستخدم للإسقاط على السطح xz والسطح yz، وتعطى المصفوفات المقابلة بالتالي:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 6.2$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 6.3$$

مثال:
اعتبر الكائن الموضح في الشكل 6.7. اوجد الإسقاط المتعامد للوجه front والجنب side في الاتجاه الموضح بالسهم.



الشكل 6.7

الحل

مشهد الواجهة المحدد هو الإسقاط على السطح xy، باستخدام المعادلة 6.1، تصبح مصفوفة النقاط كالاتى:

$$[P']_{xy} = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

- نتحصل على مشهد الجانب باستخدام الإسقاط المتعامد على السطح yz. باستخدام المعادلة 6.2 نتحصل على

$$[P']_{yz} = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

تستخدم دعوات الدوال التالية للإسقاطات المتعامدة في مكتبة التخطيط *OpenGL*

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

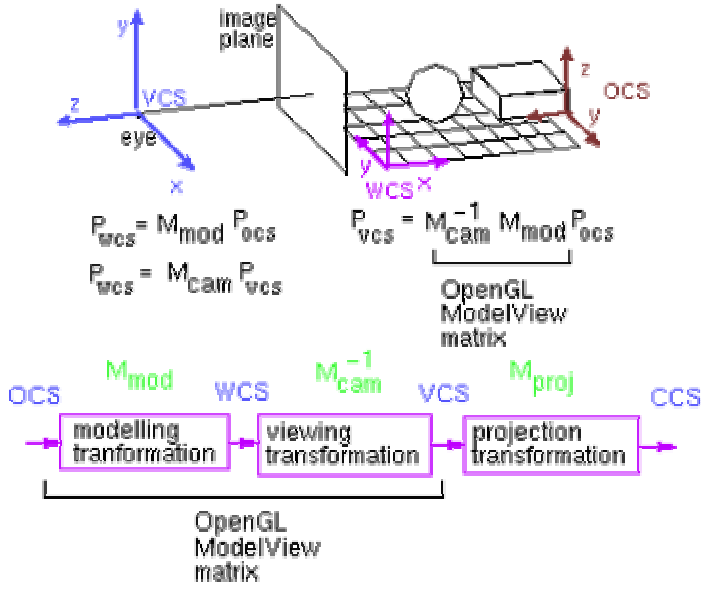
ثم يتبع بواحد من

```
glOrtho(left, right, bottom, top, near, far)
gluOrtho2D(left,right,bottom,top)
```

أعلاه يكون near > 0 و far > 0 لوضع أسطح القطع على z=-near و z=-far. إن دعوة الدالة gluOrtho2D() يشابه دعوة الدالة glOrtho() مع اخذ near=0 و far=1

6. 6. تحويلات المشاهدة Viewing Transformations

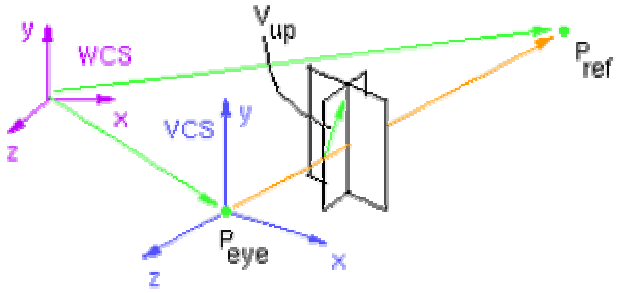
نمذجة التحويلات تعبر عن موقع الكائن في العالم. يمكننا اشتقاق تحويل المشاهدة بتحديد أولا موقع الكاميرا في العالم. يعكس هذا التحويل للتعبير عن موقع العالم بالنسبة للكاميرا ، الشكل 6.8.



الشكل 6.8: تحويلات المشاهدة

لتحديد موقع واتجاه الكاميرا قد نستخدم كما هو موضح فى الشكل 6.9:

- نقطة العين
- نقطة مرجعية
- متجه أعلى



الشكل 6.8: تحديد موقع الكاميرا

مكتبة OpenGL الإضافية توفر الدالة *gluLookAt()* لإنتاج تحويل المشاهدة.

gluLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz)

حيث *(ex,ey,ez)* تعطى نقطة العين the eye point و تعطى *(rx,ry,rz)* النقطة المرجعية أو نقطة المشاهدة the reference or 'lookat' point و *(ux,uy,uz)* يعطى المتجه الأعلى the up vector.

تستدعى الدالة الطبقات المتعددة اللاحقة postmultiplies للمصفوفة الحالية، وعليه أسهل طريقة لاستخدامها هي:

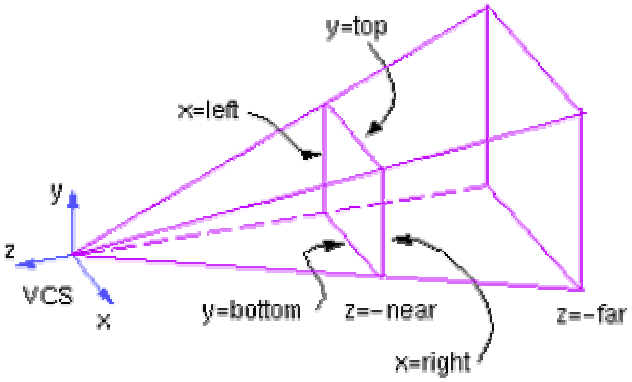
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz);
/* setup modelling transformations here */
```

6.7. أحجام المشهد View Volumes

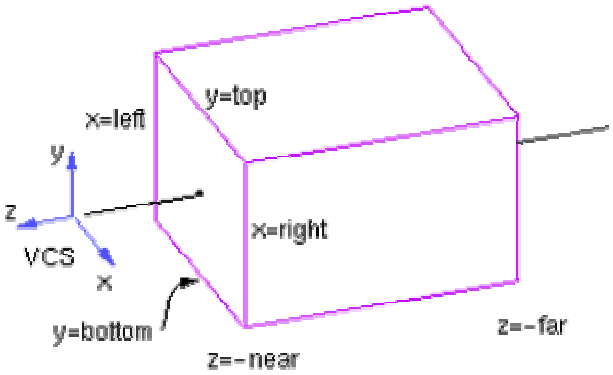
تستخدم إحجام المنظر للقيام بالعمليات التالية:

- قص الكمائن (أجزاء المشهد) التي لا تسقط في نافذة المشاهدة
- تحديد نطاق z لحسابات الرؤية

يعطى حجم المشهد المنظورى كما هو موضح في الشكل 6.9. إما حجم المنظر المتعامد فيعطى كما هو موضح في الشكل 6.10.

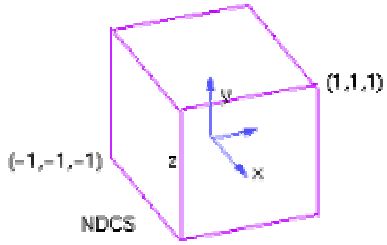


الشكل 6.9: حجم منظر الإسقاط المنظورى



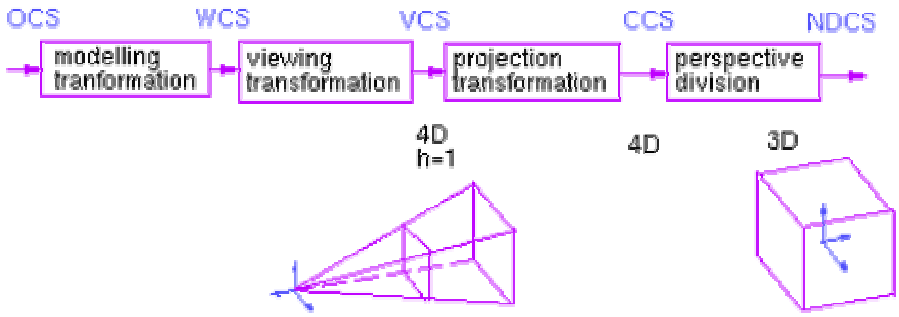
الشكل 6.10. حجم منظر الإسقاط المتعامد

يمكن القيام بعمليات القص مباشرة بالاعتماد إحجام المنظر المحددة، ولكن من ألأبسط وألا سهل استخدام إحجام المنظر المخروطية canonical view volume. يعرف نظام الإحداثيات باسم نظام إحداثيات الجهاز المعياري normalized device coordinate system or NDCS، الشكل 6.11.



الشكل 6.11. نظام الإحداثيات NDCS

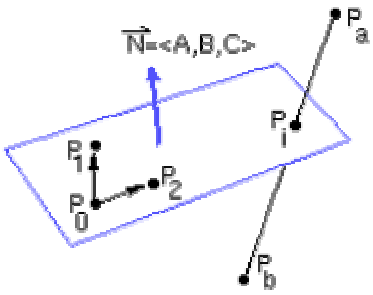
التحويل من نظام إحداثيات النظر VCS إلى نظام إحداثيات الجهاز المعياري NDCS يمكن النظر إليه كجزء من تحويلات الإسقاط، كما هو موضح فى الشكل 6.12.



الشكل 6.12. التحويل بين نظم الإحداثيات

6.8. معادلات السطح *Plane Equations*

الشكل 6.13، يوضح العوامل المختلفة للسطح والتي يمكن استخدامها لإيجاد معادلات السطح أدناه والتي تقسم إلى ثلاثة أنواع.



الشكل 6.13. عوامل السطح

- ***معادلات السطح الضمنية Implicit plane equation***

Ax + By + Cz + D = F(x,y,z)

F(x,y,z)=0

لكل النقاط على السطح الذي يعطى باستخدام: F(P)=N.P+D

- ***معادلات السطح الوسطية (البارامترية) Parametric plane equation***

السطح Plane(s,t)

Plane(s,t) = P0 + s(P1-P0) + t(P2-P0),

وذلك باعتبار كل من P0 و P1 و P2 غير مرتبطين خطيا.

بالنسبة للسطح *Plane(s,t)*

Plane(s,t) = P0 + s V1 + t V2,

حيث V1 و V2 يمثلان رأسان أساسيان.

معادلة السطح الغير مباشرة Explicit plane equation

z = -(A/C)x - (B/C)y - D/C ,

هذه المعادلة صالحة لقيم C الموجبة.

يعطى تقاطع الخط والسطح بالمعادلة:

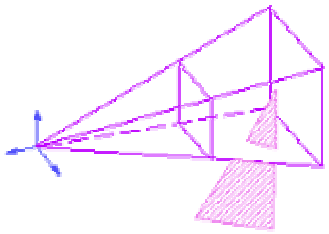
L(t) = Pa + t(Pb - Pa)

بالتعويض في معادلة السطح والحل للعامل t نتحصل على:

$$\vec{N} \bullet (P_a + t(P_b - P_a)) + D = 0$$

$$t = \frac{-D - \vec{N} \bullet P_a}{\vec{N} \bullet P_b - \vec{N} \bullet P_a} = \frac{-F(P_a)}{F(P_b) - F(P_a)}$$

6.9. القص Clipping



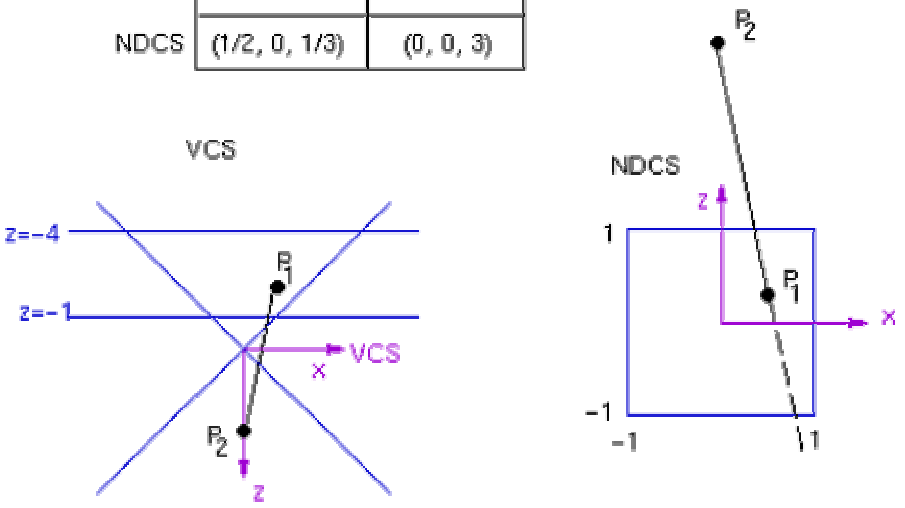
الشكل 6.14: القص في الفضاء الثلاثي الأبعاد

الشكل 6.14 يوضح القص في الإحداثيات الثلاثية الأبعاد. إن خوارزمية كوهين-سيزارلاند يمكن تمديدها للتعامل مع المناظر الثلاثية الأبعاد. يمكننا القيام بالقص في من النظم الأحداثيات VCS أو CCS أو NDCS. وإن كان هناك عيوب في استخدام NDCS كما سنذكر لاحقا في هذه الوحدة.

- القص في إحداثيات Clipping in VCS VCS
كل من خوارزمية قص الخط وخوارزمية قص المضلع تستخدم اختبارات الدخل والخرج في نصف الفضاء. بالنسبة لأحجام المناظر المتعامدة التي تعرضنا لهل سابقا إن خوارزمية كوهين-سيزارلاند للقص تعمل بنفس الطريقة في حالة الأبعاد الثلاثة مثل في حالة الأبعاد الثنائية. حيث توجد شفرات الرؤوس وتفحص للقبول أو الرفض. إن لم يوجد قبول أو رفض بديهي (مباشر)، يقص الخط مقابل احد أسطح مشاهدة الحجم الست، ويختبر مرة ثانية، وتستمر العملية.

- القص في إحداثيات NDCS
تمثل إحداثيات NDCS نظام جيد لعمليات القص حيث أن معادلات ألأسطح تحدد ببساطة وتنقى دون تغير. بالإضافة إلى ذلك، الخطوط في VCS تماثل خطوط في NDCS، وعليه يمكن حساب التقاطعات، مع اعتبار الحقيقة إن فضاء NDCS لديه اعوجاج غريب نسبة للقسمة قبل الرسم المنظورى. إن المشكلة التي قد تنتج للقص في NDCS هي فقدان إشارة المعلومات كما هو موضح في المثال الموضح فى الشكل 6.15.

	P_1	P_2
VCS	(1, 0, −2)	(0,0,2)
CCS	(1, 0, 2/3, 2)	(0,0,−6,−2)
NDCS	(1/2, 0, 1/3)	(0, 0, 3)



الشكل 6.15: القص في إحداثيات NDCS

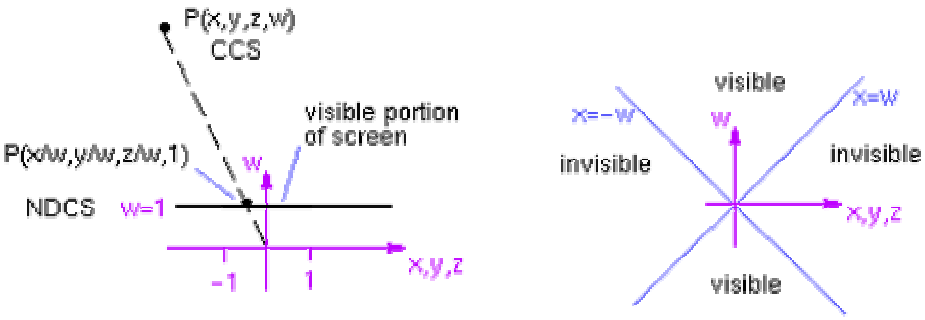
القص في إحداثيات Clipping in CCS CCS
لكي نحدد مدى القص في CSS علينا أولا النظر إلى مدى القص في NDCS

$$-1 <= x/w <= 1$$

وبعني هذا انه في CSS لدينا

$$-w <= x <= w$$

يمكن المشاهدة كنا هو موضح في الشكل 6.16.

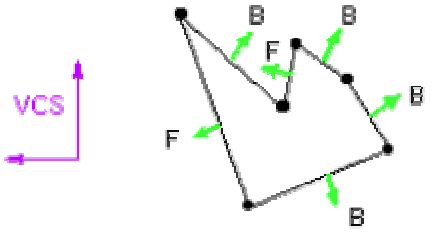


الشكل 6.11: القص في CCS

الوحدة السابعة: الرؤية و نماذج الإضاءة
Visibility and Illumination Models

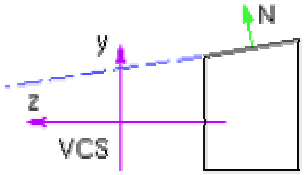
7.1. تنقية الوجه الخلفي *Back-face Culling*

تحذف تنقية الوجه الخلفي مباشرة كل المضلعات التى لا تواجه المشاهد، كما هو موضح بالشكل 7.1. يمكن استخدام تنقية الوجه الخلفي في إحداثيات VCS أو NDCS. لنستعرض أولا القص في VCS.



الشكل7.2. حذف الأوجه

في إحداثيات VCS قد تكون المحاولة الأولى لإجراء تنقية الوجه الخلفي هي مباشرة استخدام المكون z في السطح المتعامد كما يعبر عنه في VCS كما هو موضح بالشكل 7.2.. يفشل هذا ألأسلوب في كثير من الأحيان.



الشكل7.2. المحاولة المباشرة في نظام VCS

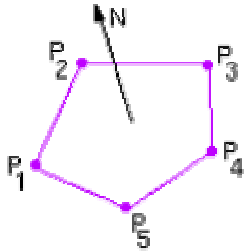
ألإستراتجية الأفضل لاستخدام تقنية الوجه الخلفي هي إنشاء معادلة السطح للمضلع والاختبار إن كان نظر المشاهد يسقط على سطح يحتوى على المضلع أو لا. باختبار سطح المشاهدة Plane(Peye)، وأن كان Plane(Peye)<0 يعنى إن نظر المشاهد أسفل السطح الذي يحتوى على المضلع وعندها تستخدم التنقية.

يمكن تلخيص التنقية في VCS بالتالي:

- أحسب متعامد السطح N=(A,B,C) ويتم معيراته
- أوجد D في معادلة السطح plane equation باستبدال اى رأس مضلع في معادلة السطح
- أحسب Plane(Peye) لتحديد مستوى الرؤية أعلى أو أسفل، ويماثل هذا مراجعة إشارة D.

لنعتبر الآن تنقية الأوجه في إحداثيات NDCS. في هذا النظام المكون z لمتعامد السطح يعكس الرؤية الحقيقية. إن كان هذا المكون موجب، يشير المتعامد بعيدا عن العين وعليه يجب تنقية المضلع. الشكل 7.3.

من الواضح للقيام بتنقية الوجه نحتاج إلى متعامد السطح. يمكن استخدام احد الطريقتان أدناه لإيجاد متعامد السطح.



الشكل7.3. إيجاد الرؤية في نظام NDCS

الطريقة الأولى:
تستخدم حاصل الضرب الموجه cross-product لحافات المضلعين. الترتيب الذي يستخدم لتخزين الرؤوس يجب أن يكون متناسق consistence. مثال لهذا، إن حفظت رؤوس المضلع بالترتيب CCW عند النظر إليها من أعلى وجه المقدمة يمكننا رؤية:

$$N = (P2 - P1) \times (P3 - P2)$$

الطريقة الثانية:
طريقة أخرى أكثر صعوبة وهي استخدام الأسطح المسقطة في yx و xz و yz. للتحديد المساحات التي يمكن استخدامها لحساب المتعامد، لنعتبر أولا حالة ثنائية الأبعاد.

7.2. تحديد الرؤية *Visibility*

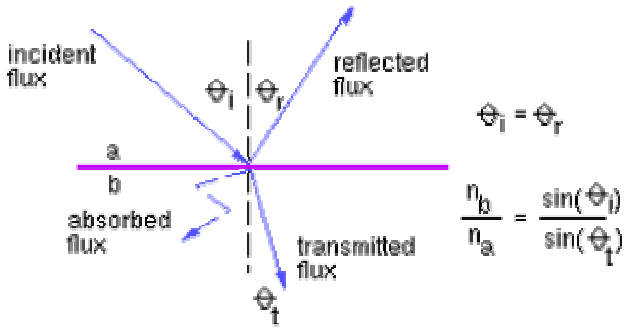
نحتاج لخوارزميات الرؤية لتحديد مكونات المشهد التي تحجبها مكونات أخرى. يمكن تصنيف هذه الخوارزميات في مجموعتين:

- خوارزميات فضاء المنظر image-space algorithms
  - تستخدم على بدائيات العرض مثل عنصر الصورة و خطوط المسح
  - تحسن الرؤية حتى مستوى دقة العارض
  - أمثلة: صادم-z و وادكين Watkin's و تتبع الشعاع ray-tracing

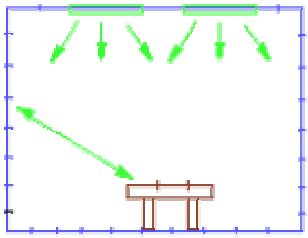
- خوارزميات فضاء الكائن object-space algorithm
  - فواصل المجال الثنائي
  - اختلافات في خوارزميات الدهان (التظليل) painter's algorithm
  - أسوا حالة: إنتاج $O(N^2)$ بدائية من N بدائية أصلية

7.3. نماذج الاستضاءة illumination Models

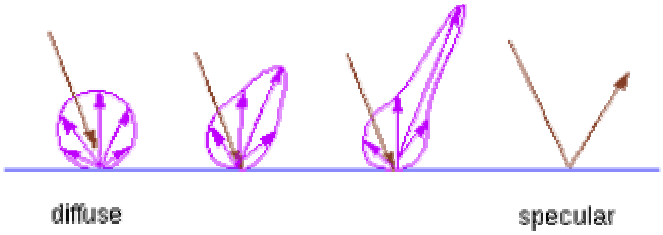
ألاستضاءة المحلية local illumination
يحدد ضوء منفرد يتقاطع مع سطح واحد



ألاستضاءة الشاملة global illumination
تعطى نماذج لتبادل الأضواء بين جميع ألسطح و نهتم هنا باستضاءة موقع معين، واتت أهمية هذا من مجالات انتقال الحرارة والتطبيقات الهندسية والعلمية



الصفات العامة لنماذج الاستضاءة المحلية Local Illumination Models هي:



- السرعة في الحساب
- تجريبي heuristic وغير كامل
- جيدة للضوء في اتجاه نقطة المشاهدة

أكثر نماذج الإضاءة المستخدمة لديها ثلاثة مكونات
الاستضاءة = المحيط + الانتشار + البريق

I = ambient + diffuse + specular

حد المحيط يسمح ببعض التحكم الشامل في شدة إضاءة المشهد، نموذجيا

I = Ia * ka

حيث Ia ثابت استضاءة المحيط يحدد مرة واحدة لكل المشهد، و ka معامل انعكاس المحيط ambient reflection coefficient وعادة يقيد بان يكون في المدى [0,1].

عادة يتصف انتشار الانعكاس بالتالي:

- انعكاس لامبرتين *Lambertian* reflection
- شدة إضاءة السطح الظاهرة لا تعتمد على زاوية المشاهدة
- توزيع الضوء المبعثر scattered light لا يعتمد على اتجاه وصول الضوء

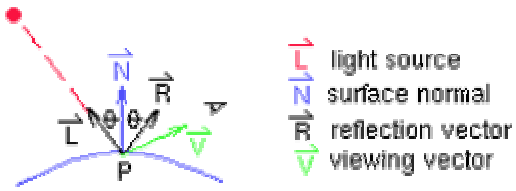
ويعطى حد الانتشار بالمعادلة التالية

Id = Ii k_diff cos(θ),

حيث

- Ii : شدة مصدر الضوء i
- k_diff: معامل انعكاس السطح
- θ الزاوية بين اتجاه الضوء ومتعامد السطح وتكون قيمة في المدى [-π/2, π/2]

العامل الأخير الذي يحدد الاستضاءة المحلية هو البريق. والشكل أدناه يوضح الوضع:



يستخدم نموذج فونق عادة لحساب البريق كالآتي:

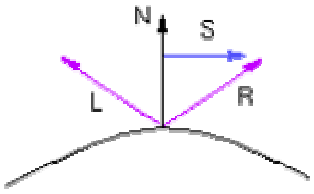
$$I_s = I_i\, k_{spec} \cos^n(\alpha)$$
$$= I_i\, K_{spec}\, (R.V)^n$$

حيث

k_spec: معامل بريق السطح

α : الزاوية بين متجه الانعكاس والمشاهدة.

عامل السطح n يستخدم لتحديد درجة خشونة السطح للسطح الأملس (المرآة) n=∞ للأسطح الخشنة جدا n=1. و تحسب R كالآتي:

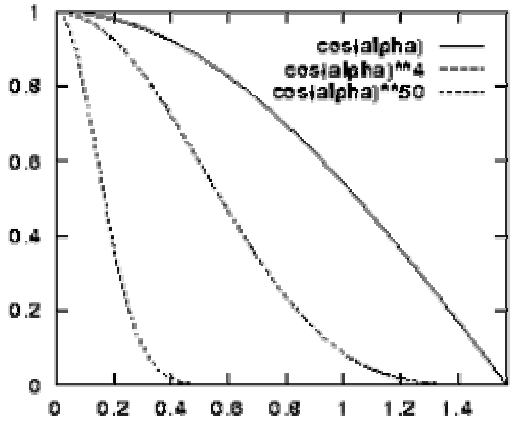


$R = (N \bullet L)\, N + S$

$S = (N \bullet L)\, N - L$

$R = 2N\,(N \bullet L) - L$

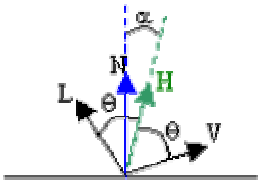
الدالة $\cos^n(\alpha)$ تبدو كالآتي:



يمكن أيضا حساب البريق باستخدام نموذج بليين Blinn الذي توافق أكثر مع نتائج التجارب، ويستخدم متجه منتصف المسافة H كالآتي:

$$I_s = I_i\, k_{spec} \cos^n(alpha)$$
$$= I_i\, K_{spec}\, (N.H)^n$$

$$H = \frac{L + V}{\| L + V \|}$$



ميزات هذا النموذج تشتمل على:
الأساس النظري
حاصل الضرب N.H لا يمكن أن يكون سالب إذا كان $N.L > 0$ و $N.V > 0$
بالنسبة لاتجاه الضوء والمسقط المتعامد قيمة H تكون ثابتة

نتحصل على النموذج الكلى للاستضاءة المحلية بضم النماذج المختلفة المذكورة أعلاه وبافتراض نموذج فونق للاستضاءة نتحصل على:

$$I = I_a\, ka + I_i\, k_{diff}\, (N.L) + I_i\, k_{spec}\, (R.V)^n$$

حيث كل من العوامل k_a و k_{diff} و k_{spec} تصف الأسطح المختلفة وتكون قيمتها في المدى من 0 إلى 1.

للتعامل مع الألوان تستخدم ثلاثة معادلات مشابهة للمعادلات أعلاه وهى:

$$I_r = I_{a\_r}\, k_{a\_r} + I_{i\_r}\, k_{diff\_r}\, (N.L) + I_{i\_r}\, k_{spec\_r}\, (R.V)^n$$
$$I_g = I_{a\_g}\, k_{a\_g} + I_{i\_g}\, k_{diff\_g}\, (N.L) + I_{i\_g}\, k_{spec\_g}\, (R.V)^n$$
$$I_b = I_{a\_b}\, k_{a\_b} + I_{i\_b}\, k_{diff\_b}\, (N.L) + I_{i\_b}\, k_{spec\_b}\, (R.V)^n$$

عند استعمال هذا النموذج يلاحظ التالي:

- عندما تكون قيمة 0 > I1 تقص I1 إلى واحد
- عندما تكون 0 > N.L تقص I1 إلى صفر أو يقلب المتعامد
- للتعامل مع مصادر ضوء متعددة، اجمع شدة كل منهم للحصول على الشدة الكلية.

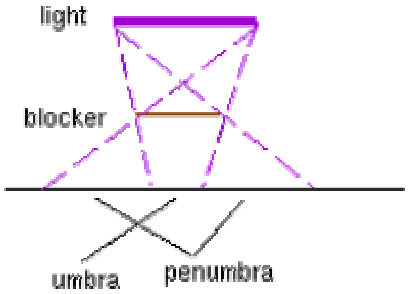
7.4. التظليل *Shading*

تستخدم في التخطيط بالحاسوب خوارزميات التظليل Shading Algorithms التالية:

- التظليل المستوى *flat shading*
  يستخدم نموذج الاستضاءة مرة واحدة للمضلع، ويستخدم هذا كلون لكل المضلع
- تظليل قورود *Gouraud shading*
  يحسب نموذج الاستضاءة عند الرؤوس، ثم يستكمل هذه الألوان عبر المضلع
- تظليل فوتق *Phong shading*
  يستكمل المتعامدات خلال عملية تحويل المسح، ثم يطبق نموذج الاستضاءة عند كل عنصر صورة.

يمكن تظليل المشهد بأحد الطرق التالية:

- الأداء الإسقاطى projective rendering
- يرسم الكائن لمرة أخرى باستخدام تحويلات تظليل إضافية
- تسويد الكائن الموجود
- يستخدم مصد طباعة stencil buffer ليمنع التظليل المزدوج `double shadows'
- تتطبع الإشعاع ray tracing
- جزء من الطريقة
- يبث أشعاع التظليل
- مساحة مصادر الضوء area light sources



تعانى خوارزميات التظليل من المشاكل التالية:

- الاعتماد على الاتجاه
- الصورة الغير ظليه silhouettes
- المشهد المشوش
- توليد متعامدات الرؤوس

7.4. نماذج الإضاءة في مكتبة OpenGL

يستدعى التالي لضبط عوامل الضوء

```
glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba );
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba );
glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba );
glLightfv(GL_LIGHT0, GL_POSITION, position);
glEnable(GL_LIGHT0);
```

المستدعيات التاليات يحددون خواص السطح لاستخدامهم فيما بعد في كل الكمائن (الأشياء) التي يتم رسمها.

```
glMaterialfv( GL_FRONT, GL_AMBIENT, ambient_rgba );
glMaterialfv( GL_FRONT, GL_DIFFUSE, diffuse_rgba );
glMaterialfv( GL_FRONT, GL_SPECULAR, specular_rgba );
glMaterialfv( GL_FRONT, GL_SHININESS, n );
```

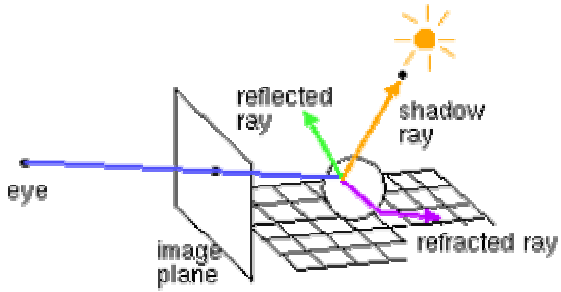
7.5. تتبع الإشعاع Ray Tracing

يستخدم اسلوب تبع الإشعاع للآتى:

لضبط المشاهدة والأشياء (الكمائن) الشفافة transparent objects

استضاءة شاملة للضوء المرآوى

استضاءة جزءية على موقع معين فى الهندسة الضوئية geometric optics



الوحدة الثامنة: المنحيات وألا سطح
Curves and Surfaces

8.1. المنحنيات الوسيطة (البارامترية) Parametric Curves

يمكن تمثيل المنحنيات والأسطح بطرق مباشرة أو غير مباشرة أو باستخدام تمثيل وسيطة (بارامترية).
إن التمثيل البارامترى لا يكون في العادة أحادى، ولتوضيح هذا اعتبر التمثيل البارامترى للخط :

$L(P_0,P_1) = P_0 + u(P_1-P_0),\ u = [0...1]$
$L(P_0,P_1) = v(P_1-P_0)/2 + (P_1+P_0)/2,\ v = [-1...1]$

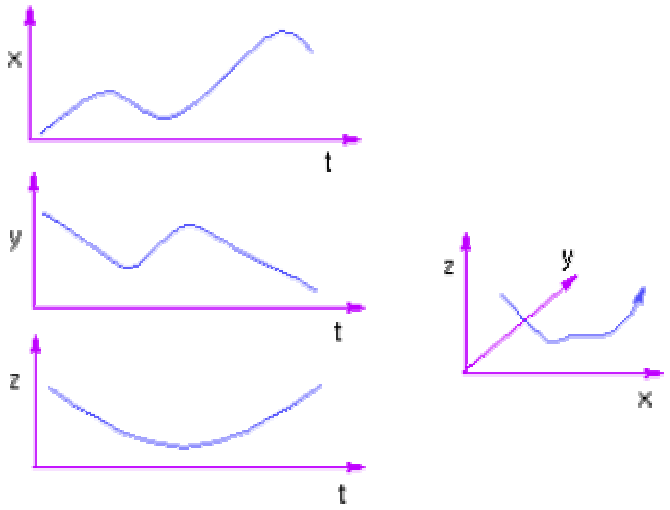
يمكن تغير العوامل لتاخد اى قيمة بين قيدين معينين. لتحويل العوامل من u = [a...b] إلى w = [0...1] يمكننا استخدام w = (u-a)/(b-a) والتي تعطى u = w(b-a) + a وعليه نتحصل على

$P(u),\ u = [a...b] \quad = \quad P(w(b-a)+a),\ w = [0...1]$

المنحنيات المكعبة cubic curves تستخدم بكثرة في التخطيط، نسبة لعدم مرونة منحنيات الترتيب الأدنى ومنحنيات الترتيب الأعلى تضيف تعقيدات عير مطلوبة. تحدد عوامل المنحى التكعيبي بالتالي:

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$$
$$z(t) = c_3t^3 + c_2t^2 + c_1t + c_0$$

في العادة يمكن اعتبار t = [0...1].



يمكن كتابة المعادلات الوسيطة (البارامترية) اعره بصورة مدمجة على النحو التالي:

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x(t) = T \cdot A$$

بطريقة مماثلة يمكننا كتابة :

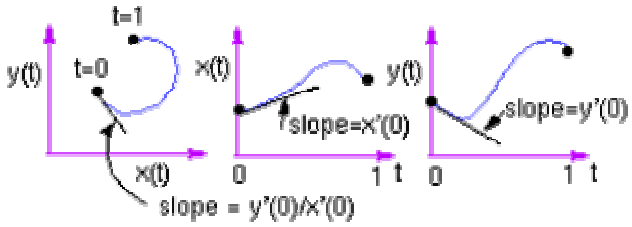
$y(t) = T\ B$
$z(t) = T\ C$

كل بعد يتم التعامل معه بصورة مستقلة عن بقية الأبعاد. وعليه يمكننا التعامل مع منحيات لديها اى عدد من الأبعاد.

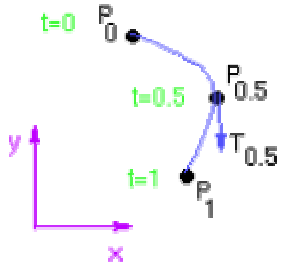
مشتقات derivatives المنحى بالنسبة إلى t يعبر عنه كالاتى:

$$x'(t) = [3t^2 \quad 2t \quad 1 \quad 0]\ A$$

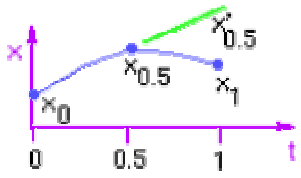
من السهل اعتبار العامل t يمثل الزمن بالتتريب لمشاهدة بعض مواصفات المنحنى. للمشتقات تفسيرات أولية في الفضاء الديكارتى للمنحنى:



مثال 1:
لنفترض بأننا نريد تحديد منحى تكعيبي بحيث يمكن للمستخدم تحديد موقع نقاط النهاية endpoints والنقطة الوسيطة midpoint بالإضافة إلى المماس عند النقطة الوسيطة.
الشكل أدناه يوضح بعض نماذج المنحنيات.



في البدء ننشئ المنحنى التكعيبي x(t). لتحقيق هذا يوحد أربع شروط constraints لابد من تحقيقهم وأربع مجاهل.



$$x(0) = [\,0\ 0\ 0\ 1\,]\ A$$

$$x(0.5) = [\,0.5^3\ 0.5^2\ 0.5^1\ 0.5^0\,]\ A$$

$$x'(0.5) = [\,3(0.5^2)\ \ 2(0.5)\ \ 1\ \ 0\,]\ A$$

$$x(1) = [\,1\ 1\ 1\ 1\,]\ A$$

or $G_x = B\ A$ where

$$G_x = \begin{bmatrix} x_0 \\ x_{0.5} \\ x'_{0.5} \\ x_1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.125 & 0.25 & 0.5 & 1 \\ 0.75 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

يمكننا الحل لإيجاد A باستخدام $A = B^{-1}x$. وعليه تكون المعادلة النهائية × على النحو التالي:

$$x(t) = T\, B^{-1}\, G_x$$

المصفوفة B^{-1} تدعى مصفوفة الأساس ويرمز إليها بالحرف M . عليه يمكننا كتابة:

$$x(t) = T\, M\, G_x$$

وفى هذه الحالة يكون لدينا

$$M = \begin{bmatrix} -4 & 0 & -4 & -4 \\ 8 & -4 & 6 & -4 \\ -5 & 4 & 2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

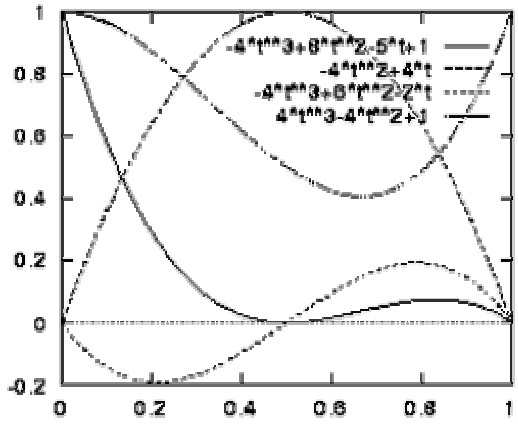
أخيرا يمكننا أيضا كتابة

$$x(t) = [\, f_1(t)\ f_2(t)\ f_3(t)\ f_4(t)\,]\ G_x$$

حيث يتحصل على الدوال $f_1(t)$... $f_4(t)$ من حاصل ضرب $T * M$. تدعى هذه الدوال دوال الأساس أو الركائز *basis functions* or *blending* ، حيث إنها تعطى الأوزان للمكونات المختلفة للمتجه الهندسي G. وفى هذا المثال هذه الأوزان هي:

$f_1(t) = -4t^3 + 8t^2 - 5t + 1$
$f_2(t) = -4t^2 + 4t$
$f_3(t) = -4t^3 + 6t^2 - 2t$
$f_4(t) = 4t^3 - 4t^2 + 1,$

حيث الدالة $f_1(t)$ الوزن بالنسبة إلى P_o و $f_2(t)$ الوزن بالنسبة إلى $P_{0.5}$ و $f_3(t)$ الوزن بالنسبة إلى $T_{0.5}$ و $f_4(t)$ الوزن بالنسبة إلى P_1. تبدو دوال الأساس على الهيئة التالية:



المنحنيات y(t) و z(t) تنشا بصورة مماثلة لإنشاء المنحنى x(t). وتبقى مصفوفة الأساس ودوال الأساس على نفس الحال فقط يتغير المتجه الهندسي. حيث يعطى المنجه الهندسى للدالة y(t) مكونات y فى P_0 و $P_{0.5}$ و $T_{0.5}$ و P_1. عادة يمكننا كتابة (التعبير) عن المنحنى كمعادلة متجهة واحدة وهى:

$$P(t) = T\, M\, G$$

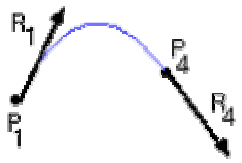
تشتمل هذه المعادلة على المعادلات الثلاث التالية:

$x(t) = T\ M\ G_x$
$y(t) = T\ M\ G_y$
$z(t) = T\ M\ G_z$

8.2. منحنيات هيرميت

مثال 2

لنعتبر منحنيات هيرميت Hermite curves والتي تحدد بنقطتين ومماسين متجهين.



لنشتق معادلة المنحنيات هيرميت باستخدام المتجه الهندسي التالي:

$$G_h = [\ P_1\ P_4\ R_1\ R_4\ ]^T$$

كما ذكر سابقا في هذه الوحدة، نعبر عن المنحنى بالتالي:

$$x(t) = T\ A_h$$
$$= T\ M_h\ G_h$$

الشروط التي سوف نستخدمها لتعريف المنحنى هي:

$x(0) = P_1 = [\ 0\ 0\ 0\ 1\ ]\ A_h$
$x(1) = P_4 = [\ 1\ 1\ 1\ 1\ ]\ A_h$
$x'(0) = R_1 = [\ 0\ 0\ 1\ 0\ ]\ A_h$
$x'(1) = R_4 = [\ 3\ 2\ 1\ 0\ ]\ A_h$

كتابه هذه الشروط في هيئة مصفوفة يعطى:

$$G_h = B_h\ A_h$$
$$A_h = (B_h)^{-1}\ G_h$$
$$x(t) = T\ A_h$$
$$= T\ (B_h)^{-1}\ G_h$$
$$= T\ M_h\ G_h$$

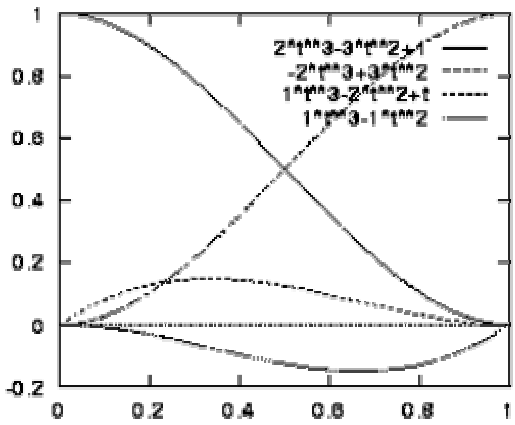
مقلوب المصفوفة B_h يعرف مصفوفة الأساس لمنحنى هيرميت

$$M_h = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

كما كان في المثال السابق، تمثل دوال الأساس الأوزان للحدود في المتجه الهندسي وتعطى بحاصل ضرب $T\ M_h$. عليه دوال الأساس لمنحنى هيرميت هي:

$f_1(t) = 2t^3 - 3t^2 + 1$
$f_2(t) = -2t^3 + 3t^2$
$f_3(t) = t^3 - 2t^2 + t$
$f_4(t) = t^3 - t^2$

تبدو دوال أساس منحنى هيرميت على الهيئة التالية:



8.3. منحنيات بيزير

مثال 3: منحنيات بيزير Bezier Curves

منحنيات بيزير تعتبر متغير من منحنيات هيرميت، حيث إنها تحدد بأربع نقاط.

المنحنى مرتبطة بشدة بمنحنى هيرميت حيث

$P_{1h} = P_1$
$P_{4h} = P_4$
$R_{1h} = 3 (P_2 - P_1)$
$R_{4h} = 3 (P_4 - P_3)$

نتحصل على مصفوفة الأساس بالتعبير عن المعادلات أعلاه في هيئة المصفوفات:

$$\begin{bmatrix} P_{1h} \\ P_{4h} \\ R_{1h} \\ R_{4h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$P_h = M_{bh} P_b$

عليه نكتب المعادلات لمنحنى بيريز على النحو التالي:

$P(t) = T M_h M_{bh} P_b$
$P(t) = T M_b P_b$

حيث:

$$M_b = \begin{bmatrix} -1 & 3 & 3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

وتكون دوال أساس بيريز كالاتى:

$P(t) = T M_b G_b$
$= f_1(t) P_1 + f_2(t) P_2 + f_3(t) P_3 + f_4(t) P_4$

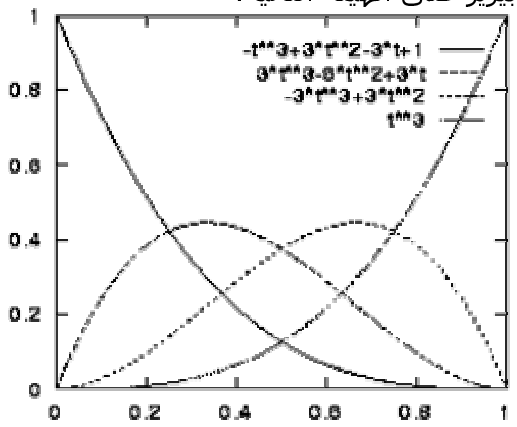
حيث

$f_1(t) = -t^3 + 3t^2 - 3t + 1$
$f_2(t) = 3t^3 - 6t^2 + 3t$
$f_3(t) = -3t^3 + 3t^2$
$f_4(t) = t^3$

أو بأسلوب أخر

$f_1(t) = (1-t)^3$
$f_2(t) = 3t\,(1-t)^2$
$f_3(t) = 3t^2\,(1-t)$
$f_4(t) = t^3$

تبدو دوال أساس منحنى بيريز على الهيئة التالية:



تستخدم خاصية الغطاء المحدب Convex hull property للتأكد من أن المنحنى لا يتعدى أبدا الغلاف المحدب الذي ينشا عبر رؤوس التحكم الأربعة. تستوفى دوال أساس بيزير خاصية الغرف المحدب تحديدا:

$0 <= f_i(t) <= 1$ for t in [0,1].
$f_1(t) + ... + f_n(t) = 1$

8.4. منحنيات بيزير من الدرجة n *Bezier Curves of degree n*

باستخدام معادلة بيرنستيين المتعددة الحدود Bernsteinn polynomials يمكننا إنشاء منحنى بيزير بدرجة اعتباطية. للمنحيات الدرجات أعلى من منحنى بيزير التي نفشت حتى الآن، نحتاج إلى المزيد من نقاط التحكم. يعطى منحنى بيزير من الدرجة n كالاتى:

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i$$
$$\text{where } B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

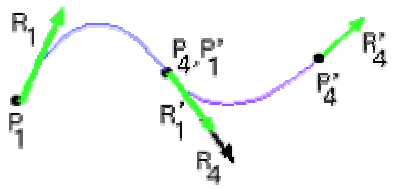
دوال الأساس متكافئو مع الحدود التي تنتج من تمديد

$$[(1-t)+t]^n$$

وذلك باستخدام نظرية التمديد الثنائي الحد binomial expansion theorem . ومن الواضح هنا لماذا يكون مجموع الحدود يساوى واحد.

8.5. منحيات هيرميت وبيزير المتقطعة *Piecewise Hermite and Bezier Curves*

مقاطع segments منحى هيرميت يمكن تربطها في هيئة مستمرة بالنا كد من نقطة نهاية منحى هي نقطه بداية المنحنى الأخر كما يجب التأكد أن متجهات المماس في هذه النقطة لديها نفس الاتجاه.



بدلالة الشكل أعلاه، هذا يعنى أن $P_1'=P_4$ و $R_1'=k\,R_4$. بالنسبة منحنى بيزير الشرط هو تتطابق أخر نقطة في المنحنى مع أول نقطة في المنحنى الثاني.

8.6. ألاستمرارية الهندسة والبارامترية *Geometric and Parametric Continuity*

تعطى الاستمرارية الهندسية بالتالي:

- G_0: المنحنيات مرتبطة
- G_1: المشتقات الأولى تناسبية عند نقاط الارتباط، مماسات المنحيات لديها نفس الاتجاه ولكن لسن ما الضرورة نفس المرتبة magnitude.
- مثلا: $C_1'(1) = (a,b,c)$ and $C_2'(0) = (k*a, k*b, k*c)$
- G_2: تتناسب المشتقات الأولى والثانية عند نقاط الارتباط

تعطى الاستمرارية البارامترية بالتالي:

- C_0: المنحنيات مرتبطة
- C_1: المشتقات ألأولى متساوية
- C_2: المشتقات الأولى والثانية متساوية
  إن اعتبرت t تمثل الزمن، يعنى هذا أن التسارع مستمر.
- C_n: المشتقات من الدرجة n متساوية

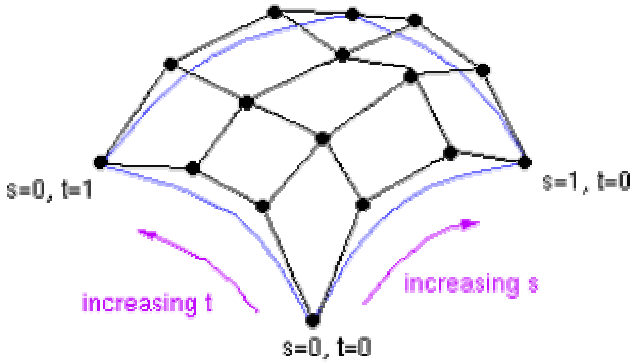
ويلاحظ هنا إن الاستمرارية البارامترية من الدرجة n تؤكد الاستمرارية الهندسية من الدرجة n ولكن العكس غير صحيح.

باعتبار منحنيات هيريت وبيزير نجد الأتي:

- نعطى تحكم محلى
- تعطى استمرارية C_1
- تستكمل بعض نقاط التحكم

8.7. الأسطح البارامترية *Parametric Surfaces*

طريقة بسيطة لتعميم المنحنيات البارامترية إلى ألسطح البارامترية هي بالسماح لنقاط التحكم للتغير لمجموعة من المنحنيات البارامترية. المثال التالي يوضح إنشاء رقعة بيزير بهذا الأسلوب.



إن اعتبرنا مقطع من منحنى بيزير يعرف مثلا بالتالي:

P(s) = S M G_bez,
S = [s^3 s^2 s 1], s in [0,1]

يمكننا الاختيار بان نجعل عناصر G_bez هي منحنيات بيزيز

$$G\_bez = \begin{bmatrix} P1(t) \\ P2(t) \\ P3(t) \\ P4(t) \end{bmatrix}$$

حيث

P1(t) = T M G1
P2(t) = T M G2
P3(t) = T M G3
P4(t) = T M G4

كل من G1 و G2 و G3 و G4 تمثل المتجهات الهندسية التي تحدد المنحنى التكعيبي البارامتيرى بدالة t لواحد من نقاط التحكم الأربعة للمنحنى الأساسي (الأصلي) P(S). عليه يوجد ست عشر قيمة نحتاجها لتحديد كل من x(s,t) و y(s,t) و z(s,t). ويكون من المفيد ليجاد أسلوب لكتابة p(S) هيئة مدموجة ومناسبة. يمكن عمل هذا كالاتى. أولا تحول المعادلات من P4(t) ... P1(t) لإنتاج التالي:

P1(t) = G1^t M^t T^t
P2(t) = G2^t M^t T^t
P3(t) = G3^t M^t T^t
P4(t) = G4^t M^t T^t

ثم عوض هذه المعادلات في المعادلة الأصلية للدالة P(S)

$$P(s) = S\,M\,G$$

$$= S\,M \begin{bmatrix} P1(t) \\ P2(t) \\ P3(t) \\ P4(t) \end{bmatrix}$$

$$= S\,M \begin{bmatrix} G1\hat{}t & M\hat{}t & T\hat{}t \\ G2\hat{}t & M\hat{}t & T\hat{}t \\ G3\hat{}t & M\hat{}t & T\hat{}t \\ G4\hat{}t & M\hat{}t & T\hat{}t \end{bmatrix}$$

$$= S\,M \begin{bmatrix} g11 & g12 & g13 & g14 \\ g21 & g22 & g23 & g24 \\ g31 & g32 & g33 & g34 \\ g41 & g42 & g43 & g44 \end{bmatrix} M\hat{}\,t\; T\hat{}\,t$$

= S M G_tensor M^t T^t

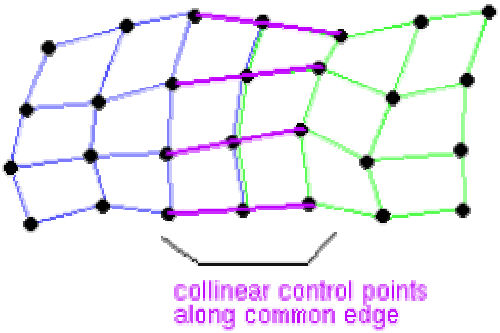
هذا يعرف حاصل ضرب تعميم المتجهة *tensor product* لتكوين السطح. وبالنسبة للمنحنيات البارامترية تمثل متجهات وعاديه يمكن كتابتها في ثلاث معادلات منفصلة وهى:

x(s,t) = S M G_x M^t T^t
y(s,t) = S M G_y M^t T^t
z(s,t) = S M G_z M^t T^t
s,t in [0,1]

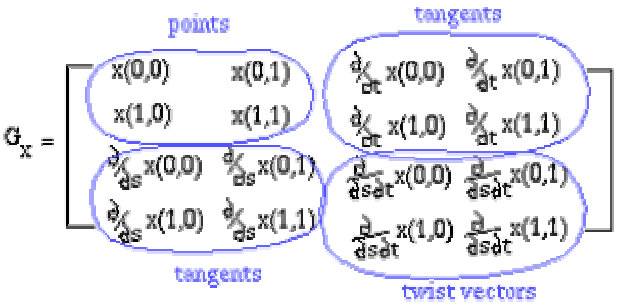
كل من G_x و G_y و G_z فى المعادلات أعلاه مصفوفة قيم متدرجة scalar حجمها 4×4

يكون لسطح بيزير الخواص التالية:

- G لديها ست عشر نقطة تحكم
- تصح خاصية الغرف المحدب
- تعطى استمرارية G1: تجعل مجموعتان من أربع نقاط تحكم في كل جانب من الحافة الشبة الخطية



الشكل أدناه يوضح سطح هيرميت

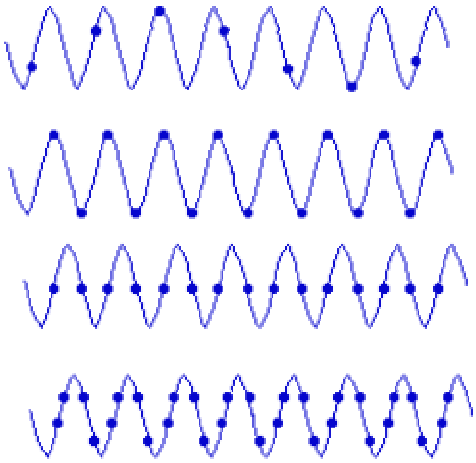


الوحدة التاسعة: مواضيع مختارة
Selected Topics

9.1. أخذ العينات *Sampling*
تنص نظرية أخذ العينات كالاتى:
يمكن استعادة الإشارة الزمنية المستمرة بالكامل من عيناتها إذا كان معدل أخذ العينات أكبر بما يقل عن مرتين عن اكبر تردد بالإشارة

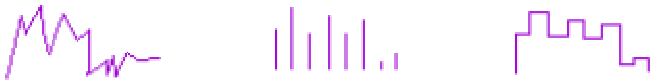
تعرف هذه النظرية بنظرية تيكويست Nyquist أو شانون Shannon. ويدعى ادني معدل اخذ عينات لاستعادة الإشارة كاملة بمعدل نيكويست *Nyquist rate*.

مثال 1:



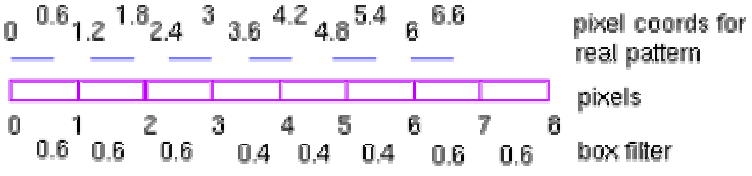
مثال 2:

الشكل أدناه يوضح عمليات أخذ العينات وإعادة الإنشاء



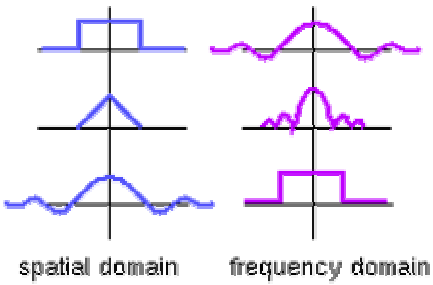
الاستراتيجيات التي تستخدم عادة لتقليل التداخل بين الإشارات هي
- الترشيح المبكر prefilter
- الترشيح المؤجل postfilter

يمثل الترشيح المبكر اخذ عينات من مساحة دون أوزان، ويتصف بالتالي:
- استخدام متوسط شدة مساحة مربع عنصر الصورة
- يستكمل بين الكائن والخلفية عند الحافات
- رقم كفاءته لديه بعض المشاكل
- ترشيح صندوق `box filter'

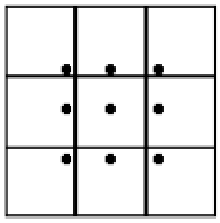


تردد أخد العينات: f
تردد الإشارة: 5f/6
تردد الخرج: f/6

الشكل أدناه يوضح عملية الترشيح في نطاق المكان spatial والتردد frequency .



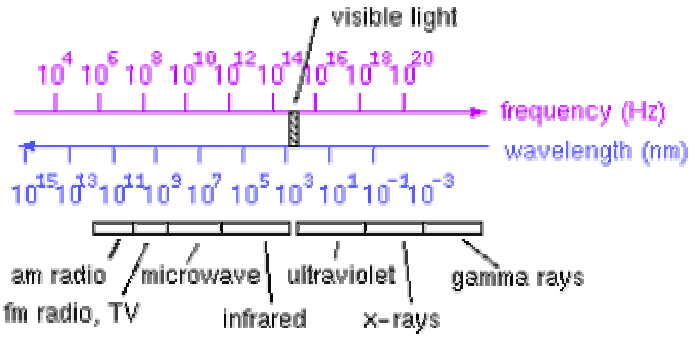
يأخذ في الترشيح المؤجل عدد من العينات لكل عنصر صورة كما هو موضح ادناه.



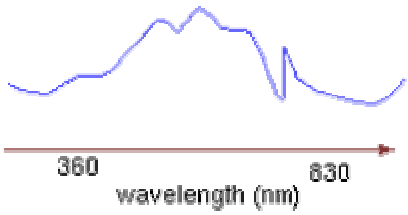
3x3 Bartlett	5x5 Bartlett
1 2 1	1 2 3 2 1
2 4 2	2 4 6 4 2
1 2 1	3 6 9 6 3
	2 4 6 4 2
	1 2 3 2 1

9.2. تمثيل الألوان *Colour Representation*

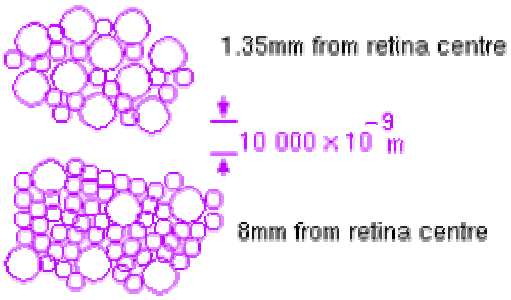
يتكون الضوء المرئي من طيف من الطاقة الكهرومغناطيسية لديها أطوال موجات في المدى 400-700 nm. طيف الموجات الكهرومغناطيسية موضح في الشكل أدناه:



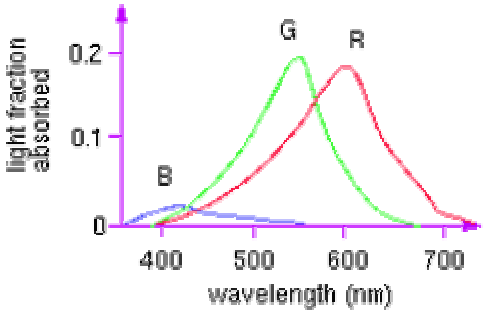
يوضح الشكل أدناه طيف الطاقة الناتجة من انعكاس الضوء من سطح اخضر.



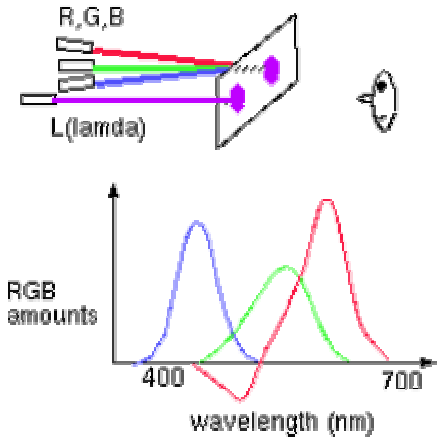
تحتوى شبكية العين على قضبان *rods* وإشكال مخروطية *cones* كما هو موضح أدناه. ألأجسام المخروطية هي مسئولة عن إدراك الألوان.



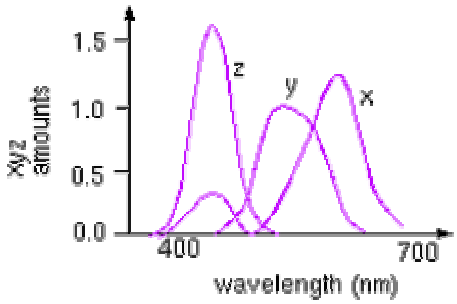
توجد ثلاثة أنواع من المخروطيات، يشار إليها بالأحرف B و G و R أو مكافئة للأحرف L و M و S بالتوالي. قمة الحساسية تكون عند حوالي 430nm و 560nm و 610nm لدى المشاهد العادي. تثار القضبان والمخروطيات بامتصاص الضوء والذي يحدث تغيرات في جهد غشاء الخلية.



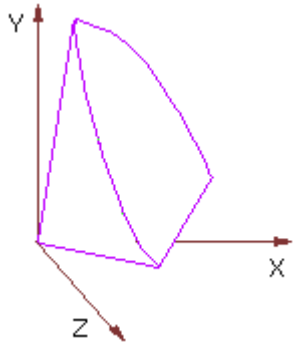
إدراكنا للألوان مرتبط بمحاكاة ثلاث أنواع من المخروطيات. حيث يتم إدراكنا للفضاء الثلاثي الأبعاد بمزج ثلاثة ألوان رئيسة مثل الأحمر R (طول الموجة، λ = 700 nm) والأخضر G (طول الموجة، λ = 546 nm) والأزرق B (طول الموجة، λ = 436 nm). وبهذه الطريقة يمكننا إنتاج إي طول موجة باستخدام مزيج محدد من B G R. قد تحدث مشكلة مثلا لا بد من إضافة اللون ألأحمر للهدف قبل الحصول على المزيج المطلوب، يوضح هذا في الرسم البياني بجعل شدة اللون ألأحمر R (في هذا المثال) تأخذ قيمة سالبة.



للحصول على التمثيل باستخدام معامل مزج موجبة فقط، حددت CIE ("Commission Internationale d'Eclairage") ثلاثة مصادر ضوء افتراضية جديدة تعرف بالأحرف x و y و z . تنتج هذه المصادر منحنيات ملائمة matching curves موجبة.



لنفترض الآن بأنه لدينا طيف (لون) ونود إيجاد الكميات X و Y و Z المقابلة لهذا اللون. يمكننا إيجاد هذه الكميات بتكامل حاصل ضرب القدرة الطيفية spectral power مع كل من منحنيات الملائمة الثلاثة لطل أطوال الموجات wavelengths. أوزان X و Y و Z لفضاء XYZ المحدد فى CIE موضحة أدناه.

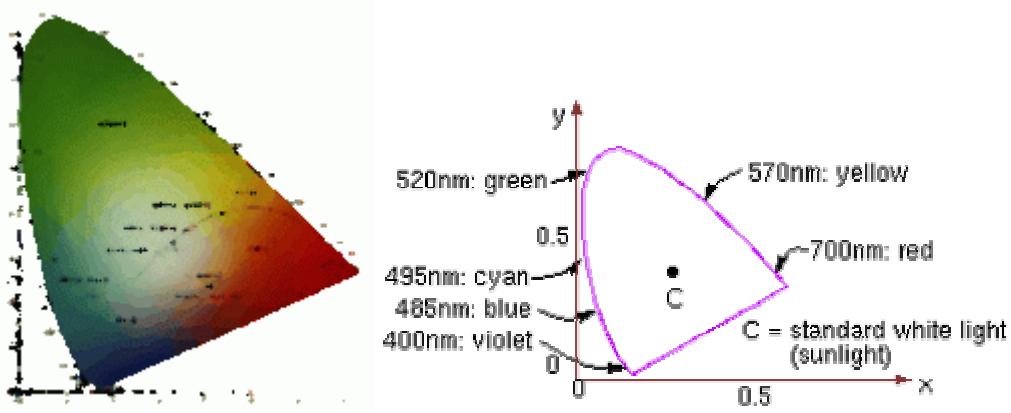


عادة من السهل التعامل مع قضاء الألوان الثنائي الأبعاد 2D colour space. يتم هذا في العادة بإسقاط فضاء الألوان الثلاثي الأبعاد 3D colour space على السطح X+Y+Z=1 لإيجاد مخطط لونية *CIE* *chromaticity diagram*. CIE . تحدد الإسقاطات بالتالي:

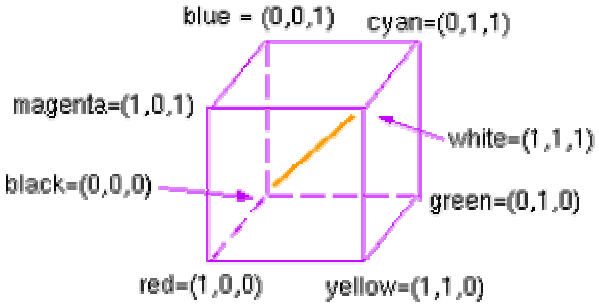
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z} = 1 - x - y$$

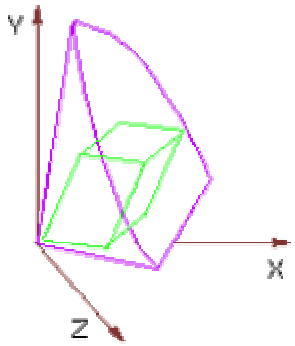
يبدو المخطط اللوني كالاتى:



نموذج الألوان الجمعي المستخدم في التخطيط بالحاسوب يمثل بمكعب اللون RGB colour cube RGB ، حيث تمثل R و G و B أللألوان المنتجة عن الفسفور النقى ألأحمر والأخضر والأزرق على التوالى.



الشكل أدناه يوضح موقع مكعب الألوان في فضاء CIE للألوان.



الجدول أدناه يلخص خواص أنواع أحبار الطباعة الأربعة الأساسية.

dye colour	*absorbs*	*reflects*
cyan	red	blue and green
magenta	green	blue and red
yellow	blue	red and green
black	all	none

لإنتاج الأزرق، يمزج الأحبار ذات اللون cyan و اللون magenta، حيث كلاهما يعكسان الأزرق احدهم يمتص الأخضر والأخر يمتص الأحمر. يجعل هذا الأمر عملية تحويل ألوان العارض لألوان طباعة مكافئة عملية تحتاج إلى مجهود وتحدى. يستخدم الحبر الأسود للتأكد من إمكانية إنتاج اسود بدرجة عالية، ويشار إليه باستخدام K. وعليه تستخدم الطابعات نموذج CMYK.

الوحدة العاشرة: أمثلة تطبيقية لاستخدام مكتبة التخطيط OpenGL

10.1. أمثلة لبرمجيات OpenGL

كما ذكر في الوحدة الثانية توجد عدد من مكتبات المرئي التي توفر ابتدائيات رسم لتكوين المشاهد المختلفة، أكثر هذه المكتبات استخداما على مستوى الحاسوب الشخصى هى مكتبة OpenGL فى التى تتصف بالتالى:

- لا يعتمد البدائيات على العتاد حيث يمكن استخدامها على الحواسيب الشخصية PC ومحطات العمل workstations والحواسيب الفائقة القدرة Supercomputer
- يعتمد على نظام التشغيل ويدعم نظم التشغيل التالية: Windows NT, Windows 2000, Windows XP Linux and Unix
- يمكن أن ينفذ باستخدام البرامج مثال المعالج أو العتاد مثل لوحة تسريع التخطيط accelerated graphics card إن وجدت.
- يمكن استخدامه مع لغات البرمجة التالية: C, C++, Ada, Fortran, Java
- يدعم أداء المضلع polygon rendering التخطيط البنيوي texture mapping و وعدم استخدام الاسم المستعار anti-aliasing
- لا يدعم متابعة الإشعاع ray tracing و أداء الحجم volume rendering

في هذه الوحدة سنعطى بعد الأمثلة لاستخدام مكونات مكتبة OpenGL مع لغة البرمجة C. حيث ستعطى ملفات المصدر source files والترويسة header files لكل مثال. يمكن للدارس تطبيق هذه الأمثلة على جهاز حاسوب شخصي يستخدم نظام التشغيل يونكس/لينكس Unix/Linux أو Windows. يمكن ترجمة ملفات المصدر باستخدام مترجمات البرامج cc أو gcc لمستخدمي اليونكس أو Visual Studio C++ version 6.0 لمستخدمي windows. في كل الحاجات تنفيذ هذه التطبيقات يحتاج إلى تنصيب مكتبة OpenGL و برامج GLUT في الجهاز (ملفات الترويسة للترجمة وكائن المكتبة لتنفيذ البرنامج).

لمستخدمي يونكس/لينكس، أدناه مثال لبرامج الترجمة والربط *makefiles* التي قد تستخدم لترجمة وربط برامج المثال الأول **SamplDraw**، للأمثلة الأخرى اجري التعديلات الضرورية لبرنامج makefile وذلك بإضافة اسم برنامج المصدر c. في سطر أمر الترجمة compiler command line بالترتيب الصحيح. وبرنامج الترجمة والربط هي:

- برنامج الترجمة والربط للإستخدام مع برنامج الترجمة cc

```
INCDIRS = -I/usr/openwin/include -I/usr/local/include
LIBDIRS = -L/usr/openwin/lib -L/usr/local/lib

CC = gcc
CFLAGS = -g $(INCDIRS)
LIBS = -lX11 -lXi -lXmu -lglut -lGL -lGLU -lm


SimpleDraw: SimpleDraw.o
    $(CC) $(CFLAGS) -o SimpleDraw $(LIBDIRS) SimpleDraw.c $(LIBS)
```

- برنامج الترجمة والربط للإستخدام مع برنامج الترجمة cc

```
INCDIRS = -I/usr/openwin/include -I/usr/local/include
LIBDIRS = -L/usr/openwin/lib -L/usr/local/lib

CC = gcc
CFLAGS = -g $(INCDIRS)
LIBS = -lX11 -lXi -lXmu -lglut -lGL -lGLU -lm


SimpleDraw: SimpleDraw.o
    $(CC) $(CFLAGS) -o SimpleDraw $(LIBDIRS) SimpleDraw.c $(LIBS)
```

في حالة استخدام Visual C++ هناك احتمال كبير إن تكون مكتبة OpenGL قد تم تنصيبها مع تصيب المترجم. الصفحة الإلكترونية الرسمية والتي يمكن منها تحميل أخر إصدارات البرنامج موجودة على العنوان http://www.xmission.com/~nate/glut.html. بعد التحميل تتحصل على الملفات glut32.dll و glut32.lib و glut.h. يجب تنصيب هذه البرامج على جهاز الحاسوب الشخصي، لمزيد من التفاصيل راجع الموقع أعلاه.

10.2. برنامج SimpleDraw

برنامج Simple Draw لشفرة C يوضح استخدام مكتبة التخطيط OpenGL. يوضح البرنامج طريقة رسم النقاط points والخطوط lines و إزالة الخطوط line strips وحلقات الخطوط line loops. البرنامج يتكون من ملفين مصدر وهما SimpleDraw.c و SimpleDraw.h.

المطلوب من الدارس التالي:

1. ترجم وشغل البرنامج، استخدم مسطرة المسافة space bar للتنقل بين المناظر الخمسة. اختبر شفرة المصدر واوجد كيف يتم رسم اى من هذه المناظر الخمسة فى وتيرة routine ***drawScence()***
2. أزيل علامات التعليق uncomment من تجميعه الشفرة في ***initRendering()*** ثم اعد ترجمة وتشغيل البرنامج، (أزيل التعليق من التجميعية ألأولى أولا ثم من التجميعيتان). كيف يوثر هذا على النتائج؟ وكيف تبدو مناظر التخطيط على الحاسوب؟
3. تفهم عملية المنادة الراجعة التي تضبط بمنادة كل من ***glutKeyboardFunc(), glutReshapeFunc(), glutDisplayFunc().***
   تفهم كيف يستجيب البرنامج لمسطرة المسافة و مفتاح الخروج escape key.
4. لا يستخدم هذا البرنامج الصد المذدوج double buffering. ما معنى هذا. ما هى وظيفة دعوة **glutPostRedisplay()** بعد تغير الحالة.

ملف SimpleDraw.c

```
/*
 * SimpleDraw.c
 *
 * Example program illustrating a simple use
 * of OpenGL to draw straight lines in 2D, and
 * to draw overlapping triangles in 3D.
 *
 * Author: Samuel R. Buss
 *
 * Software accompanying the book
 *          3D Computer Graphics: A Mathematical Introduction with OpenGL,
 *          by S. Buss, Cambridge University Press, 2003.
 *
 * Software is "as-is" and carries no warranty.  It may be used without
 *   restriction, but if you modify it, please change the filenames to
 *   prevent confusion between different versions.
 * Bug reports: Sam Buss, sbuss@ucsd.edu.
 * Web page: http://math.ucsd.edu/~sbuss/MathCG
 *
 * Version 1.1.  Updated September 28, 2003.
 *    1.1. 9/28/30: Added glEnable(GL_DEPTH_TEST).  Thanks to Rob Wilkens.
 *    1.0. Original version released Spring 2003.
 *
 */

#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>    // OpenGL Graphics Utility Library
#include "SimpleDraw.h"
```

```
// These variables control the current mode
int CurrentMode = 0;
const int NumModes = 5;

// These variables set the dimensions of the rectanglar region we wish to view.
const double Xmin = 0.0, Xmax = 3.0;
const double Ymin = 0.0, Ymax = 3.0;

// glutKeyboardFunc is called below to set this function to handle
//          all "normal" ascii key presses.
// Only space bar and escape key have an effect.
void myKeyboardFunc( unsigned char key, int x, int y )
{
    switch ( key ) {

    case ' ':                                                          //
Space bar
            // Increment the current mode, and tell operating system screen
needs redrawing
            CurrentMode = (CurrentMode+1)%NumModes;
            glutPostRedisplay();
            break;

    case 27:                                                           //
"27" is theEscape key
            exit(1);

    }
}


/*
 * drawScene() handles the animation and the redrawing of the
 *          graphics window contents.
 */
void drawScene(void)
{
    // Clear the rendering window

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Set drawing color to white
    glColor3f( 1.0, 1.0, 1.0 );

    switch (CurrentMode)
    {

    case 0:
            // Draw three points
            glBegin(GL_POINTS);
            glVertex2f( 1.0, 1.0 );
            glVertex2f( 2.0, 1.0 );
            glVertex2f( 2.0, 2.0 );
            glEnd();
            break;

    case 1:
    case 2:
    case 3:
```

```
            if ( CurrentMode==1 ) {
                // Draw lines in GL_LINES mode
                glBegin( GL_LINES );
            }
            else if ( CurrentMode==2 ) {
                // Draw lines in GL_LINE_STRIP mode
                glBegin( GL_LINE_STRIP );
            }
            else {
                // Draw lines in GL_LINE_LOOP mode
                glBegin( GL_LINE_LOOP );
            }
            glVertex2f( 0.5, 1.0 );
            glVertex2f( 2.0, 2.0 );
            glVertex2f( 1.8, 2.6 );
            glVertex2f( 0.7, 2.2 );
            glVertex2f( 1.6, 1.2 );
            glVertex2f( 1.0, 0.5 );
            glEnd();
            break;

    case 4:                         // Overlapping triangles
            glBegin( GL_TRIANGLES );
            glColor3f( 1.0, 0.0, 0.0 );
            glVertex3f( 0.3, 1.0, 0.5 );
            glVertex3f( 2.7, 0.85, 0.0 );
            glVertex3f( 2.7, 1.15, 0.0 );

            glColor3f( 0.0, 1.0, 0.0 );
            glVertex3f(2.53, 0.71, 0.5 );
            glVertex3f(1.46, 2.86, 0.0 );
            glVertex3f(1.2, 2.71, 0.0 );

            glColor3f( 0.0, 0.0, 1.0 );
            glVertex3f(1.667, 2.79, 0.5);
            glVertex3f(0.337, 0.786, 0.0);
            glVertex3f(0.597, 0.636, 0.0);
            glEnd();
    }

    // Flush the pipeline.  (Not usually necessary.)
    glFlush();

}

// Initialize OpenGL's rendering modes
void initRendering()
{

    glEnable ( GL_DEPTH_TEST );

    // Uncomment out the first block of code below, and then the second
block,
    //            to see how they affect line and point drawing.
/*
    // The following commands should cause points and line to be drawn larger
    //       than a single pixel width.
    glPointSize(8);
    glLineWidth(5);
```

```
*/

/*
    // The following commands should induce OpenGL to create round points
and
    //      antialias points and lines.  (This is implementation dependent
unfortunately).
    glEnable(GL_POINT_SMOOTH);
    glEnable(GL_LINE_SMOOTH);
    glHint(GL_POINT_SMOOTH_HINT, GL_NICEST);  // Make round points, not
square points
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);             // Antialias the
lines
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
*/

}

// Called when the window is resized
//          w, h - width and height of the window in pixels.
void resizeWindow(int w, int h)
{
    double scale, center;
    double windowXmin, windowXmax, windowYmin, windowYmax;

    // Define the portion of the window used for OpenGL rendering.
    glViewport( 0, 0, w, h );      // View port uses whole window

    // Set up the projection view matrix: orthographic projection
    // Determine the min and max values for x and y that should appear in the
window.
    // The complication is that the aspect ratio of the window may not match
the
    //              aspect ratio of the scene we want to view.
    w = (w==0) ? 1 : w;
    h = (h==0) ? 1 : h;
    if ( (Xmax-Xmin)/w < (Ymax-Ymin)/h ) {
            scale = ((Ymax-Ymin)/h)/((Xmax-Xmin)/w);
            center = (Xmax+Xmin)/2;
            windowXmin = center - (center-Xmin)*scale;
            windowXmax = center + (Xmax-center)*scale;
            windowYmin = Ymin;
            windowYmax = Ymax;
    }
    else {
            scale = ((Xmax-Xmin)/w)/((Ymax-Ymin)/h);
            center = (Ymax+Ymin)/2;
            windowYmin = center - (center-Ymin)*scale;
            windowYmax = center + (Ymax-center)*scale;
            windowXmin = Xmin;
            windowXmax = Xmax;
    }

    // Now that we know the max & min values for x & y that should be visible
in the window,
    //              we set up the orthographic projection.
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
```

```
    glOrtho( windowXmin, windowXmax, windowYmin, windowYmax, -1, 1 );

}


// Main routine
// Set up OpenGL, define the callbacks and start the main loop
int main( int argc, char** argv )
{
    glutInit(&argc,argv);

    // The image is not animated so single buffering is OK.
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH );

    // Window position (from top corner), and size (width and hieght)
    glutInitWindowPosition( 20, 60 );
    glutInitWindowSize( 360, 360 );
    glutCreateWindow( "SimpleDraw - Press space bar to toggle images" );

    // Initialize OpenGL as we like it..
    initRendering();

    // Set up callback functions for key presses
    glutKeyboardFunc( myKeyboardFunc );                    // Handles
"normal" ascii symbols
    // glutSpecialFunc( mySpecialKeyFunc );          // Handles "special"
keyboard keys

    // Set up the callback function for resizing windows
    glutReshapeFunc( resizeWindow );

    // Call this for background processing
    // glutIdleFunc( myIdleFunction );

    // call this whenever window needs redrawing
    glutDisplayFunc( drawScene );

    fprintf(stdout, "Press space bar to toggle images; escape button to
quit.\n");

    // Start the main loop.  glutMainLoop never returns.
    glutMainLoop(  );

    return(0);      // This line is never reached.
}
```

SimpleDraw.h ملف

```
/*
 * SimpleDraw.h
 *
 * Author: Samuel R. Buss
 *
  */

// Function prototypes for SimpleDraw.c

void myKeyboardFunc( unsigned char key, int x, int y );

void drawScene(void);
```

```
void initRendering();
void resizeWindow(int w, int h);
```

10.2. برنامج SimpleAmin

برنامج SimpleAnim لشفرة C يوضح استخدام مكتبة التخطيط OpenGL. يقوم البرنامج بدورات ثلاثة مثلثات متشابكة overlapping. البرنامج يتكون من ملفين مصدر وهما SimpleAmin.c و SimpleAmin.h.

المطلوب من الدارس التالي:

1. ترجم وشغل البرنامج. حاول متحكمات لوحة المفاتيح keyboard controls. بالضغط على "r" يبدءا وينتهي تحريك الرسم animation. بالضغط على "s" تتحصل على خطوة واحدة من الحركة. الأسهم ألا علي والأسفل تتحكم في سرعة الدوران.
2. لاحظ مشكلة الاسم المستعار aliasing مع حافات المثلثات. لعمل هذا كبر النافذة لمشاهدة أفضل. ثم قلل سرعة الدوران حتى تلف المثلثات ببطء. على حافات المثلثات ترى حافات خشنة غير سوية أو حركة عناصر الصورة.
3. هذا البرنامج يستخدم المصد المزدوج double buffering لتحسين الحركة. أختبر هذا بجعل البرنامج يستخدم مصد واحد. لاستخدام وصد واحد غير العامل **GLUT_DOUBLE** إلى **GLUT_SINGLE** وأضف علامة التعليق comment منادة **glutSwapBuffers()**. كيف تبدو الحركة في حالة استخدام مصد واحد .
4. بالقرب من بداية الوتيرة drawScence() الزاوية الحالية مثبتة على أقل من 360^0 ما هو الخطاء الذي سوف يحدث إن أزيلت من الشفرة. حاول إجراء هذا التغير وحدد الخطاء. (للملاحظة بسرعة اجعل المحاكي يعمل بوتيرة عالية لفترة زمنية قصيرة ثم عمل بوتيرة منخفضة، قد تحتاج للتشغيل لفترة طويلة حتى تصبح أخطاء الفيضان/الانحسار overflow/underflow errors ملموسة.
5. تفهم عمل كل من **glTranslatef** and **glRotatef**.

ملف SimpleAmin.c

```
/*
 * SimpleAnim.c
 *
 * Example program illustrating a simple use
 * of OpenGL to animate motion.  Creates rotating
 * overlapping triangles.
 *
 * Author: Samuel R. Buss
 *
 * Software accompanying the book
 *          3D Computer Graphics: A Mathematical Introduction with OpenGL,
 *          by S. Buss, Cambridge University Press, 2003.
 *
 * Software is "as-is" and carries no warranty.  It may be used without
 *   restriction, but if you modify it, please change the filenames to
 *   prevent confusion between different versions.
 * Bug reports: Sam Buss, sbuss@ucsd.edu.
 * Web page: http://math.ucsd.edu/~sbuss/MathCG
 *
 * USAGE:
 *    Press "r" key to toggle (off and on) running the animation
 *    Press "s" key to single-step animation
 *    The up arrow key and down array key control the
 *                  time step used in the animation rate.  Each key
 *                  press multiplies or divides the times by a factor
 *                  of sqrt(2).
 *    Press ESCAPE to exit.
```

```
 *
 */

#include <math.h>               // For math routines (such as sqrt & trig).
#include <stdio.h>
#include <stdlib.h>             // For the "exit" function
#include <GL/glut.h>    // OpenGL Graphics Utility Library
#include "SimpleAnim.h"

int RunMode = 1;        // Used as a boolean (1 or 0) for "on" and "off"

// The next global variable controls the animation's state and speed.
float CurrentAngle = 0.0f;                      // Angle in degrees
float AnimateStep = 3.0f;                       // Rotation step per update

// These variables set the dimensions of the rectanglar region we wish to view.
const double Xmin = 0.0, Xmax = 3.0;
const double Ymin = 0.0, Ymax = 3.0;

// glutKeyboardFunc is called below to set this function to handle
//          all "normal" key presses.
void myKeyboardFunc( unsigned char key, int x, int y )
{
    switch ( key ) {
    case 'r':
            RunMode = 1-RunMode;            // Toggle to opposite value
            if ( RunMode==1 ) {
                    glutPostRedisplay();
            }
            break;
    case 's':
            RunMode = 1;
            drawScene();
            RunMode = 0;
            break;
    case 27:        // Escape key
            exit(1);
    }
}

// glutSpecialFunc is called below to set this function to handle
//          all "special" key presses.  See glut.h for the names of
//          special keys.
void mySpecialKeyFunc( int key, int x, int y )
{
    switch ( key ) {
    case GLUT_KEY_UP:
            if ( AnimateStep < 1.0e3) {                 // Avoid overflow
problems
                    AnimateStep *= sqrt(2.0);           // Increase the angle
increment
            }
            break;
    case GLUT_KEY_DOWN:
            if (AnimateStep>1.0e-6) {           // Avoid underflow problems.
                    AnimateStep /= sqrt(2.0);   // Decrease the angle increment
            }
            break;
    }
```

```
}

/*
 * drawScene() handles the animation and the redrawing of the
 *            graphics window contents.
 */
void drawScene(void)
{
    // Clear the rendering window
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   if (RunMode==1) {
            // Calculate animation parameters
      CurrentAngle += AnimateStep;
            if ( CurrentAngle > 360.0 ) {
                  CurrentAngle -= 360.0*floor(CurrentAngle/360.0);       //
Don't allow overflow
            }
    }

    // Rotate the image
    glMatrixMode( GL_MODELVIEW );                  // Current matrix affects
objects positions
    glLoadIdentity();                                    // Initialize to the
identity
    glTranslatef( 1.5, 1.5, 0.0 );                       // Translate
rotation center from origin
    glRotatef( CurrentAngle, 0.0, 0.0, 1.0 );          // Rotate through
animation angle
    glTranslatef( -1.5, -1.5, 0.0 );                     // Translate
rotation center to origin

    // Draw three overlapping triangles of different colors
    glBegin( GL_TRIANGLES );
    glColor3f( 1.0, 0.0, 0.0 );
    glVertex3f( 0.3, 1.0, 0.5 );
    glVertex3f( 2.7, 0.85, 0.0 );
    glVertex3f( 2.7, 1.15, 0.0 );

    glColor3f( 0.0, 1.0, 0.0 );
    glVertex3f(2.53, 0.71, 0.5 );
    glVertex3f(1.46, 2.86, 0.0 );
    glVertex3f(1.2, 2.71, 0.0 );

    glColor3f( 0.0, 0.0, 1.0 );
    glVertex3f(1.667, 2.79, 0.5);
    glVertex3f(0.337, 0.786, 0.0);
    glVertex3f(0.597, 0.636, 0.0);
    glEnd();

    // Flush the pipeline, swap the buffers
   glFlush();
   glutSwapBuffers();

    if ( RunMode==1 ) {
            glutPostRedisplay();  // Trigger an automatic redraw for animation
    }

}
```

```
// Initialize OpenGL's rendering modes
void initRendering()
{
   glShadeModel( GL_FLAT );    // The default value of GL_SMOOTH is usually
better
   glEnable( GL_DEPTH_TEST );         // Depth testing must be turned on
}

// Called when the window is resized
//          w, h - width and height of the window in pixels.
void resizeWindow(int w, int h)
{
    double scale, center;
    double windowXmin, windowXmax, windowYmin, windowYmax;

    // Define the portion of the window used for OpenGL rendering.
    glViewport( 0, 0, w, h );      // View port uses whole window

    // Set up the projection view matrix: orthographic projection
    // Determine the min and max values for x and y that should appear in the
window.
    // The complication is that the aspect ratio of the window may not match
the
    //              aspect ratio of the scene we want to view.
    w = (w==0) ? 1 : w;
    h = (h==0) ? 1 : h;
    if ( (Xmax-Xmin)/w < (Ymax-Ymin)/h ) {
            scale = ((Ymax-Ymin)/h)/((Xmax-Xmin)/w);
            center = (Xmax+Xmin)/2;
            windowXmin = center - (center-Xmin)*scale;
            windowXmax = center + (Xmax-center)*scale;
            windowYmin = Ymin;
            windowYmax = Ymax;
    }
    else {
            scale = ((Xmax-Xmin)/w)/((Ymax-Ymin)/h);
            center = (Ymax+Ymin)/2;
            windowYmin = center - (center-Ymin)*scale;
            windowYmax = center + (Ymax-center)*scale;
            windowXmin = Xmin;
            windowXmax = Xmax;
    }

    // Now that we know the max & min values for x & y
    //              that should be visible in the window,
    //              we set up the orthographic projection.
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   glOrtho( windowXmin, windowXmax, windowYmin, windowYmax, -1, 1 );

}


// Main routine
// Set up OpenGL, define the callbacks and start the main loop
int main( int argc, char** argv )
{
    glutInit(&argc,argv);
```

```
    // We're going to animate it, so double buffer
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );

    // Window position (from top corner), and size (width% and hieght)
    glutInitWindowPosition( 10, 60 );
    glutInitWindowSize( 360, 360 );
    glutCreateWindow( "SimpleAnim" );

    // Initialize OpenGL as we like it..
    initRendering();

    // Set up callback functions for key presses
    glutKeyboardFunc( myKeyboardFunc );                    // Handles
"normal" ascii symbols
    glutSpecialFunc( mySpecialKeyFunc );              // Handles "special"
keyboard keys

    // Set up the callback function for resizing windows
    glutReshapeFunc( resizeWindow );

    // Call this for background processing
    // glutIdleFunc( myIdleFunction );

    // call this whenever window needs redrawing
    glutDisplayFunc( drawScene );

    fprintf(stdout, "Arrow keys control speed.  Press \"r\" to run,  \"s\" to
single step.\n");

    // Start the main loop.  glutMainLoop never returns.
    glutMainLoop(  );

    return(0);      // This line is never reached.
}
```

SimpleAnim.h ملف

```
/*
 * SimpleAnim.h
 *
 * Author: Samuel R. Buss
 *
 * Software accompanying the book
 *          3D Computer Graphics: A Mathematical Introduction with OpenGL,
 *          by S. Buss, Cambridge University Press, 2003.
 *
 * Software is "as-is" and carries no warranty.  It may be used without
 *   restriction, but if you modify it, please change the filenames to
 *   prevent confusion between different versions.
 * Bug reports: Sam Buss, sbuss@ucsd.edu.
 * Web page: http://math.ucsd.edu/~sbuss/MathCG
 */

// Function prototypes for SimpleAnim.cpp

void myKeyboardFunc( unsigned char key, int x, int y );
void mySpecialKeyFunc( int key, int x, int y );
```

```
void drawScene(void);

void initRendering();
void resizeWindow(int w, int h);
/*
 * SimpleDraw.h
 *
 * Author: Samuel R. Buss
 *
  */

// Function prototypes for SimpleDraw.c

void myKeyboardFunc( unsigned char key, int x, int y );

void drawScene(void);

void initRendering();
void resizeWindow(int w, int h);
```

10.3. برنامج Solar

برنامج Solar بشفرة C يوضح استخدام مكتبة التخطيط OpenGL. يقوم البرنامج بتمثيل حركة نظام شمسي مكون من الشمس وككوب واحد وقمر واحد. البرنامج يتكون من ملفين مصدر وهما Solar.c و Solar.h.

المطلوب من الدارس التالي:

1. ترجم وشغل البرنامج. حاول متحكمات لوحة المفاتيح keyboard controls. في البداية البرنامج يظهر نجعل الكوكب ولكنه لا يدور ولكنه دائما يواجه الشمس, استخدام الأسهم في لوحة المفاتيح لمشاهدة الحركة. أوضح كيف تعمل شفرة الحركة.
2. اجعل البرنامج يستخدم مصد واحد كما فعلنا في التطبيق السابق ولاحظ التأثير.
3. أنشئ نظاما شمسيا أكثر تعقيدا بإضافة كواكب وأقمار أخرى

ملف solar.c

```
/*
 * Solar.c
 *
 * Program to demonstrate how to use a local
 * coordinate method to position parts of a
 * model in relation to other model parts.
 *
 * Draws a simple solar system, with a sun, planet and moon.
 * Based on sample code from the OpenGL programming guide
 *          by Woo, Neider, Davis.  Addison-Wesley.
 *
 * Author: Samuel R. Buss
 *
 * Software accompanying the book
 *          3D Computer Graphics: A Mathematical Introduction with OpenGL,
 *          by S. Buss, Cambridge University Press, 2003.
 *
 * Software is "as-is" and carries no warranty.  It may be used without
 *   restriction, but if you modify it, please change the filenames to
 *   prevent confusion between different versions.
 * Bug reports: Sam Buss, sbuss@ucsd.edu.
 * Web page: http://math.ucsd.edu/~sbuss/MathCG
 *
 * USAGE:
```

```
 *    Press "r" key to toggle (off and on) running the animation
 *    Press "s" key to single-step animation
 *    The up arrow key and down array key control the
 *                  time step used in the animation rate.  Each key
 *                  press multiplies or divides the times by a factor
 *                  of two (2).
 *    Press ESCAPE to exit.
 *
 */


#include "Solar.h"
#include <stdlib.h>

#include <GL/glut.h>    // OpenGL Graphics Utility Library

static GLenum spinMode = GL_TRUE;
static GLenum singleStep = GL_FALSE;

// These three variables control the animation's state and speed.
static float HourOfDay = 0.0;
static float DayOfYear = 0.0;
static float AnimateIncrement = 24.0;  // Time step for animation (hours)

// glutKeyboardFunc is called below to set this function to handle
//          all normal key presses.
static void KeyPressFunc( unsigned char Key, int x, int y )
{
    switch ( Key ) {
    case 'R':
    case 'r':
            Key_r();
            break;
    case 's':
    case 'S':
            Key_s();
            break;
    case 27:        // Escape key
            exit(1);
    }
}

// glutSpecialFunc is called below to set this function to handle
//          all special key presses.  See glut.h for the names of
//          special keys.
static void SpecialKeyFunc( int Key, int x, int y )
{
    switch ( Key ) {
    case GLUT_KEY_UP:
            Key_up();
            break;
    case GLUT_KEY_DOWN:
            Key_down();
            break;
    }
}


static void Key_r(void)
```

```
{
    if ( singleStep ) {                 // If ending single step mode
            singleStep = GL_FALSE;
            spinMode = GL_TRUE;             // Restart animation
    }
    else {
            spinMode = !spinMode;       // Toggle animation on and off.
    }
}

static void Key_s(void)
{
    singleStep = GL_TRUE;
    spinMode = GL_TRUE;
}

static void Key_up(void)
{
   AnimateIncrement *= 2.0;                 // Double the animation time
step
}

static void Key_down(void)
{
   AnimateIncrement /= 2.0;                 // Halve the animation time step

}

/*
 * Animate() handles the animation and the redrawing of the
 *          graphics window contents.
 */
static void Animate(void)
{
    // Clear the redering window
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   if (spinMode) {
            // Update the animation state
      HourOfDay += AnimateIncrement;
      DayOfYear += AnimateIncrement/24.0;

      HourOfDay = HourOfDay - ((int)(HourOfDay/24))*24;
      DayOfYear = DayOfYear - ((int)(DayOfYear/365))*365;
            }

    // Clear the current matrix (Modelview)
   glLoadIdentity();

    // Back off eight units to be able to view from the origin.
   glTranslatef ( 0.0, 0.0, -8.0 );

    // Rotate the plane of the elliptic
    // (rotate the model's plane about the x axis by fifteen degrees)
    glRotatef( 15.0, 1.0, 0.0, 0.0 );

   // Draw the sun        -- as a yellow, wireframe sphere
    glColor3f( 1.0, 1.0, 0.0 );
   glutWireSphere( 1.0, 15, 15 );
```

```
    // Draw the Earth
     // First position it around the sun
     //              Use DayOfYear to determine its position
    glRotatef( 360.0*DayOfYear/365.0, 0.0, 1.0, 0.0 );
    glTranslatef( 4.0, 0.0, 0.0 );
    glPushMatrix();                                          // Save matrix
state
     // Second, rotate the earth on its axis.
     //              Use HourOfDay to determine its rotation.
     glRotatef( 360.0*HourOfDay/24.0, 0.0, 1.0, 0.0 );
     // Third, draw the earth as a wireframe sphere.
    glColor3f( 0.2, 0.2, 1.0 );
    glutWireSphere( 0.4, 10, 10);
    glPopMatrix();                                       // Restore matrix state

     // Draw the moon.
     //       Use DayOfYear to control its rotation around the earth
     glRotatef( 360.0*12.0*DayOfYear/365.0, 0.0, 1.0, 0.0 );
    glTranslatef( 0.7, 0.0, 0.0 );
    glColor3f( 0.3, 0.7, 0.3 );
    glutWireSphere( 0.1, 5, 5 );

     // Flush the pipeline, and swap the buffers
    glFlush();
    glutSwapBuffers();

     if ( singleStep ) {
             spinMode = GL_FALSE;
     }

     glutPostRedisplay();          // Request a re-draw for animation purposes

}

// Initialize OpenGL's rendering modes
void OpenGLInit(void)
{
    glShadeModel( GL_FLAT );
    glClearColor( 0.0, 0.0, 0.0, 0.0 );
    glClearDepth( 1.0 );
    glEnable( GL_DEPTH_TEST );
}

// ResizeWindow is called when the window is resized
static void ResizeWindow(int w, int h)
{
    float aspectRatio;
     h = (h == 0) ? 1 : h;
     w = (w == 0) ? 1 : w;
     glViewport( 0, 0, w, h );      // View port uses whole window
     aspectRatio = (float)w/(float)h;

     // Set up the projection view matrix (not very well!)
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 60.0, aspectRatio, 1.0, 30.0 );

     // Select the Modelview matrix
```

```
   glMatrixMode( GL_MODELVIEW );
}


// Main routine
// Set up OpenGL, hook up callbacks, and start the main loop
int main( int argc, char** argv )
{
    // Need to double buffer for animation
    glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );

    // Create and position the graphics window
   glutInitWindowPosition( 0, 0 );
   glutInitWindowSize( 600, 360 );
   glutCreateWindow( "Solar System Demo" );

    // Initialize OpenGL.
   OpenGLInit();

    // Set up callback functions for key presses
    glutKeyboardFunc( KeyPressFunc );
    glutSpecialFunc( SpecialKeyFunc );

    // Set up the callback function for resizing windows
   glutReshapeFunc( ResizeWindow );

    // Callback for graphics image redrawing
   glutDisplayFunc( Animate );

    // Start the main loop.  glutMainLoop never returns.
    glutMainLoop(  );

   return(0);                    // Compiler requires this to be here. (Never
reached)
}
```

ملف Solar.h

```
/*
 * Solar.h
 *
 * Author: Samuel R. Buss
 *
 * Software accompanying the book
 *          3D Computer Graphics: A Mathematical Introduction with OpenGL,
 *          by S. Buss, Cambridge University Press, 2003.
 *
 * Software is "as-is" and carries no warranty.  It may be used without
 *   restriction, but if you modify it, please change the filenames to
 *   prevent confusion between different versions.
 * Bug reports: Sam Buss, sbuss@ucsd.edu.
 * Web page: http://math.ucsd.edu/~sbuss/MathCG
 */

void OpenGLInit(void);
```

```
static void Animate(void );
static void Key_r(void );
static void Key_s(void );
static void Key_up(void );
static void Key_down(void );
static void ResizeWindow(int w, int h);

static void KeyPressFunc( unsigned char Key, int x, int y );
static void SpecialKeyFunc( int Key, int x, int y );
```

المراجع

1. Sam Buss. 3D Computer Graphics: A mathematical approach with OpenGL, Cambridge University Press, 2003.

2. E. Angel, Interactive Computer Graphics – A top down approach with OpenGL, , 2nd edition, Addison Wesley, 2000

3. Donald Hearn and M. Pauline Baker Computer Graphics (C Version),, Prentice Hall, 1997.

4. J. D. Foley et al, Computer Graphics: Principles and practice., 2nd edition, Addison Wesley, 1996

5. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics, Principles and Practice, Second Edition,Addison-Wesley, 1990.

6. Vera B. Anamd, Computer Graphics and Geometric Modeling for Engineers, John Willey, 1983