

الفصل الرابع

الهندسة التحليلية المستوية و الفراغية

سوف نتكلم في هذا الفصل عن بعض المفاهيم الأساسية العامة التي يجب أن تكون مُلم بها و هي تجميع لمعلومات رياضية و استخدام هذه المفاهيم في برمجة الألعاب .

نقطة ستقيمات ..

ي

يات

ط (س ، ص)

الهندسية في الأبعاد الثلاثة (س ، ص ، ع) .

النقطة

نقطة هي

Point

موقع لمجسم .

نها كمُتجهه vector كما سنعلم فيما بعد ، و بالعامية يمكن أن نقول أن مثلا المنزل يقع

في النقطة (س ، ص ، ع) بالنسبة للعالم الذي فيه إما افتراضي كما بالألعاب أو واقعي .

و يمكن تمثيل أو تعريف النقطة برمجياً بعدة طرق و لكل طريق ميزة و عيب .

ن

سى ++

ياكل structures صفوف (نات - نات) classes

يات الثلاثة (س ، ص ، ع) و التي تخص

يل النقطة بالمصفوفات و هو أننا لا نستطيع أن نستخدم التحميل

زائد للمعاملات overload operators بحيث نستطيع أن نستخدم مثلاً علامة الضرب x في ضرب

تخدام ا دوال بها و يجب أن تعلم أن المصفوفات هي

كل من أشكال هياكل البيانات و تكوينها Data Structures و المصفوفات وكما تلاحظ فان لها عيوب

ة .)

السى++ دراسة جيدة أليس كذلك ؟ !) .

ا باستخدام المصفوفات أو إحدى هياكل البيانات التي تدعم السى++

ل + ه أو تحمليه كما يدعي

تخدام المعامل - لعملية طرح نقطتين أيضا و

overloading

هكذا .

// Defining A Point .

```
1- float Point[2];
2- struct Point
3- بداية هيكل النقطة {
4- float x,y,z;
5- Point operator+(Point point)
6- {
7- Point temp={point.x+x, point.y+y,point.z+z};
8- return temp;
9- }
10- }; // نهاية هيكل النقطة
```

في السطر ١ قمنا بتعريف مصفوفة مكونة من ثلاث عناصر و هي تمثل الثلاث إحداثيات للنقطة x,y,z . و من الممكن لتخصيص مثلا الاحداثي السيني قيمة للنقطة نقوم بكتابة

```
Point[0]=10;
```

ملحوظة :

ترتيب عناصر المصفوفة تبدأ بالرقم ٠ و ليس ١ .

ود سى ، السى ++ فهنا قمنا بتعريف أنواع من البيانات الجديدة غير وفه التي تدعمها اللغة أساسا مثل تمثيل الاعداد الصحيحة و الاعداد الكسرية .. الخ و قمنا باستخدام الهياكل structures و من الممكن استخدام الكائنات – الصفوف – الفئات Classes في تمثيل النقطة . ماذا نحتاج لكي نعرف النقطة !؟

ثلاث إحداثيات (x,y,z) لذا قمنا بتعريف ثلاث متغيرات من نوع أعداد كسرية في السطر ٤ و قمنا بتعريف معامل دالة operator function لتفعيل عملية الجمع.

ن ي ا

الإحداثيات .

ملحوظة:

النقطة

ط x, y ;

ستوي plane

الواقعة في الفراغ space فلها ثلاث إحداثيات x, y, z .

مفاهيم أساسية

١. أنواع تمثيل الأشكال الهندسية

سنعلم أن الأشكال الهندسية هي عبارة عن معادلات ما .. و لكل معادلة شكل الدائرة لها معادلة و

الكرة لها معادلة و هكذا ...

و قام علماء الرياضيات بتصنيف هذه المعادلات إلى فئتين

١. المعادلات الصريحة (التمثيل الصريح) implicit form

٢. المعادلات البارامترية (التمثيل البارمترى) . parameteric form

و كمثال بسيط – علي هذين التمثيل سنري كيف نمثل الخط المستقيم .

٢. تعريف المستوي و الفراغ

طح يتكون من مجموعة غير منتهية من النقط و سوف نري كيف نمثل المستوي جبرياً

ستوي بتمثيله بقطعة منه تشبه السبورة أو صورة ورقة أو لوحة

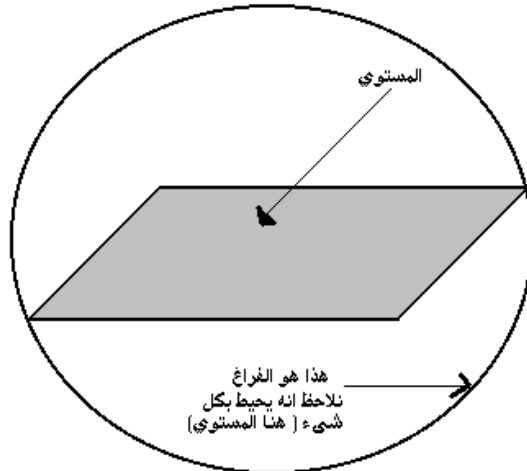
ي

ي ..

و عبارة عن مستوي أي يرسمون علي مستوي . الفراغ هو مجموعة غير منتهية من النقطة

املة لجميع النقط و المستقيمت و المستويات و المجسمات الموجودة في الكون أو العالم World .. و

سوف نقربه و نقول العالم الافتراضي للعبة .

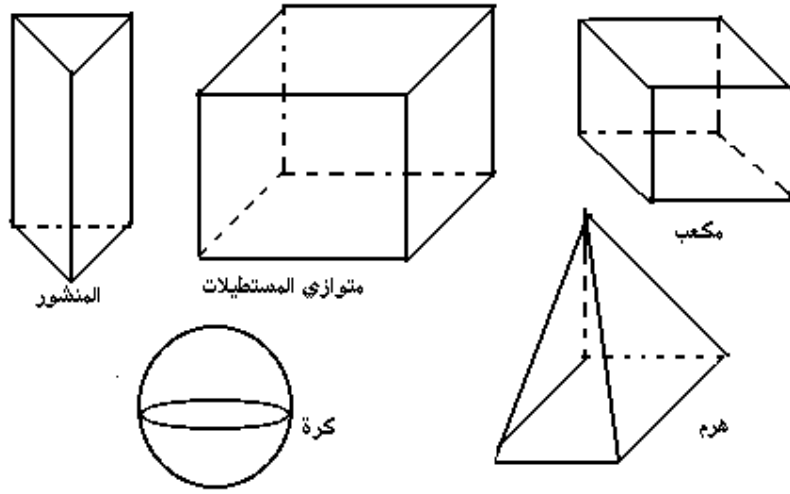


٣. تعريف المجسم

المجسم هو مجموعة جزئية غير منتهية من الفراغ محدودة بسطح أو أكثر يفصل

داخل الجسم عن خارجه مثل المكعب و متوازي المستطيلات و المنشور و الهرم...

أشكال بعض المجسمات



نلاحظ وجود خطوط منقطة و هي تدل علي أن هذا الجزء غير منظور أو لا تستطع أن تراه أي خطوط مخفية.

و جدير بالذكر لمبرمجي الرسومات و مبرمجي المحركات الثلاثية الأبعاد :

ذه

ن

"

الطريقة Back Face Removal أي إزالة السطح الخلفي."

خاص الثلاثية

ي عبارة عن مجموعة من المثلثات و لكن

لما لم نستخدم الأشكال الأخرى مثل متوازي الأضلاع أو شبه المنحرف ... الخ من الأشكال الهندسية ؟!

و ذلك لان المثلث يتميز بهذه الخصائص

١. حيث أن أي ثلاث نقط تكون مستوي لذا اخترها المثلث

٢. في صناعة المحركات أو في برمجة الرسومات مثل برمجة برنامج 3D Studio Max نحتاج إلي

سمي الحذف Clipping و سوف نعلم عندما نتطرق إلي هذه العملية في كتبنا الأخرى التي

تحدث عن برمجة الرسومات و برمجة الألعاب سوف أن نري أن المثلثات من السهل تطبيق عملية

الحذف عليها علي عكس الأشكال الهندسية الأخرى .

٣. حيث أن المثلث

الخط المستقيم

Line - Ray

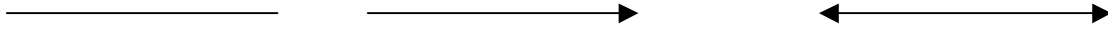
ه خصائص منها انه يمتد إلي ما

ي المصطلحات عندما

عاع Ray

ن الخط المستقيم في الهندسة المحوسبة (الهندسة باستخدام الحاسب) فهو يُسمى شعاع ray بدلاً

من المؤلف خط line.



خط مستقيم شعاع (له طول و مقدار) قطعة مستقيمة

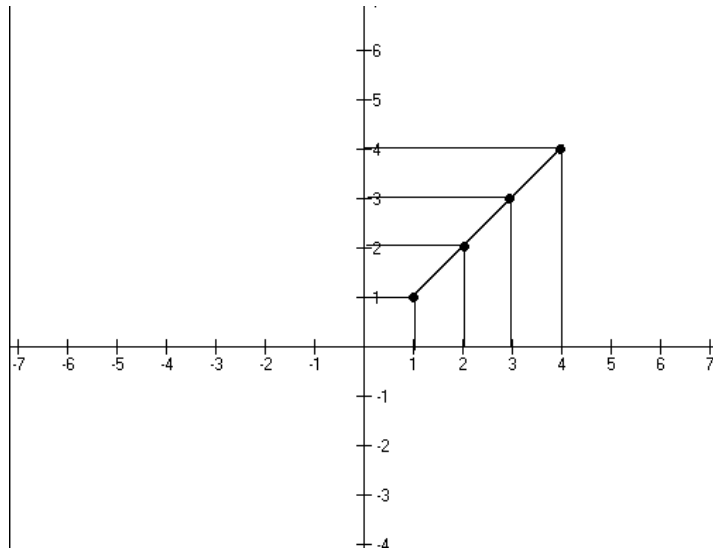
سمي ray ي برمجة الألعاب

ي line

غالباً و لكن معظم المؤلفين يستخدم Line .

الخط

المستقيم .



و كما موضح بالصورة لقد تم تعريف عدة نقطة و قمنا برسمها ابتداء من النقطة التي تقع في المستوي (

س - ص) و هي (١،١) و حتى النقطة (٤ ، ٤) و بعد ذلك قمنا بتوصيل جميع النقط اصبح لدينا خط

مستقيم .

معادلة الخط المستقيم في بعدين

لقد قام علماء الرياضيات بتعريف عدة أشكال لمعادلة الخط المستقيم و هم ثلاثة أنواع

١. المعادلة العامة لتعريف خط مستقيم

٢. معادلة الجزء المقطوع من المحاور للخط مع وجود ميل الخط .

٣. معادلة النقطة مع وجود ميل الخط .

المعادلة العامة لخط مستقيم في بعدين فقط (س،ص) هي :

$$أس + ب ص + ج = ٠$$

$$ax + by + c = 0$$

حيث أ،ب،ج و ما ينظرهما بالإنجليزي ثوابت عددية .

معادلة الجزء المقطوع من المحور (ص) مع وجود ميل للخط هي :

$$ص = أس + ب$$

$$y = m \cdot x + b$$

حيث أ هو ميل الخط المستقيم ، ب هو الجزء المقطوع من المحور الصادي .

حساب ميل الخط المستقيم و الجزء المقطوع من المحور ص

هل تلاحظ في الصورة أن هناك ميل للخط المستقيم ؟

بلي - أرى الميل ، و لكن كيف احسب مقدار ميل هذا الخط ؟!

فلنعتبر أن هذا الخط يمثل دبابة مثل و تميل بمقدار معين و نريد أن نحسب ميلها كيف ؟

ميل الخط المستقيم = مقدار التغير في المحور الصادي

مقدار التغير في المحور السيني

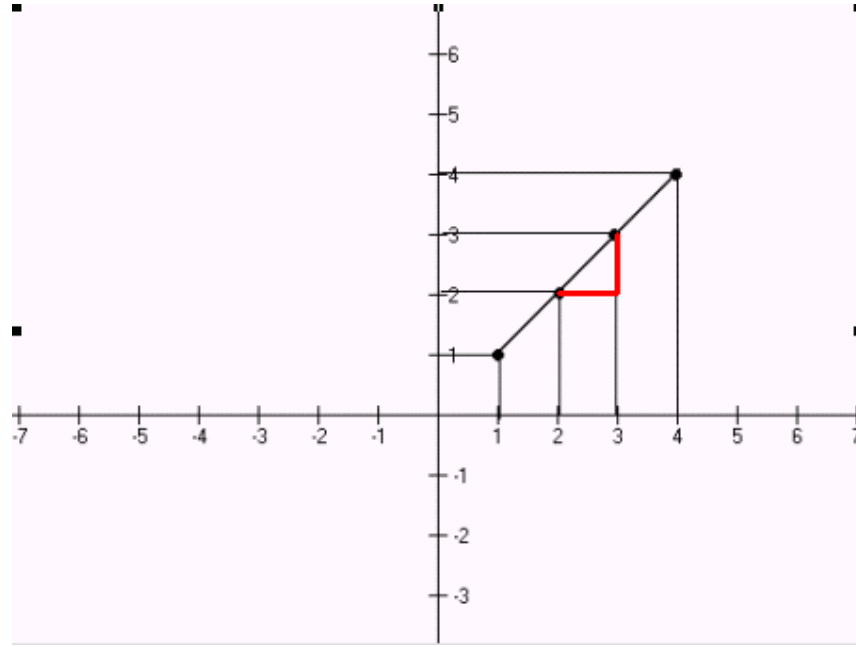
$$\text{Slope} = dy / dx$$

و يعتبر مقدار ما تغير في المحور الصادي dx عبارة عن

$$ص٢ - ص١ ، y2 - y1$$

و بالنسبة للمحور السيني نفس الشيء . و بقسمة المقدارين ينتج ميل الخط هذا جبرياً و لكن هندسيا انظر

إلى الصورة القادمة .



انظر إلي المثلث الأحمر مقدار التغير علي الصادي هو الفرق بين أي نقطتين علي المحور الصادي مثل $3 - 2 = 1$ و مقدار التغير علي المحور السيني هو الفرق بين أي نقطتين علي المحور السيني $3 - 2 = 1$ أيضاً ..

إذن الميل يساوي $1 = 1$.

و نلاحظ أن الجزء المقطوع من المحور الصادي y هو 0 و لو أردت معرفته قم بمد الخط سوف تلاحظ انه يقطع المحور في نقط الأصل $origine$ و هي الصفر .
و كمعلومة إضافية إذا كنا نريد أن نحسب مقدار ما يميله الخط بالدرجات نحسب
ظاهر = الميل

ظاهر = 1 إذن الزاوية هـ = 45 درجة

إذن الخط أو كما مثلناه بالدبابة تميل بزاوية مقدارها 45 درجة .

هناك بعض الملاحظات علي ميل خط المستقيم و يمكن استنتاج خصائص للخط.
مثال :

إذا كان هناك خطين مثلاً و فرضنا ميل الأول m_1 و ميل الآخر m_2 و كان

ميل الأول * ميل الثاني = -1 فإن

الخطين متعامدين أي الزاوية بينهم 90 درجة .

و من ذلك نستطيع أن نقول أن معادلة الجزء المقطوع مع وجود ميل هي

$$ص = أس + ب$$

و كمثال :

ص = ٥س + ٣ هذه معادلة خط مستقيم ميله = ٥ و الجزء المقطوع من المحور ص هو ٣ الوحدة (متر ، سم .. الخ) .

و المعادلة الثالثة هي معادلة خط مستقيم بتعريف نقطة و ميل

$$و هي (س - س١) = م (ص - ص١)$$

حيث م هي ميل الخط ، (س١ ، ص١) هي نقطة علي الخط .

و كمثال المعادلة :

$$(س - ٥) = ٤ (ص - ٣) .$$

بعد القسمة علي المقدار (ص - ٣) نجد أن

$$\frac{(س - ٥) = ٤}{(ص - ٣)}$$

همم هذه المعادلة مألوفة بالنسبة لي ! بالتأكيد هي الميل = التغير في ص علي التغير في س .

إذن المستقيم يقع عن النقطة (٥ ، ٣) و بميل يساوي ٤ .

و بذلك قد انهينا التحدث عن المستقيمات في المستوي (في بعدين فقط) و معني ذلك انك تستطيع بهذه

المعلومات استخدامها في الألعاب الثنائية الأبعاد و ليست الثلاثية.

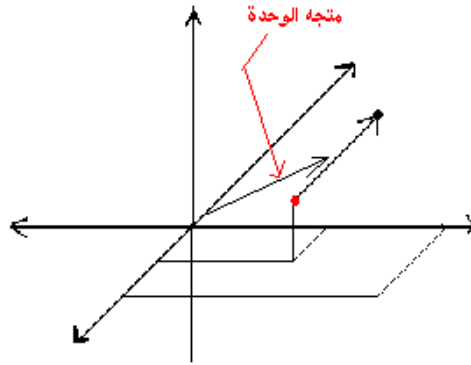
تعريف خط مستقيم في الفراغ

كما رأيت لقد قمنا بتعريف خط مستقيم في المستوي و قمنا بعرض معادلاته .. ولكن هذا التمثيل في

الألعاب الثنائية الأبعاد و لكن ماذا في الألعاب الثلاثية الأبعاد (في الفراغ) .. سوف نقوم بتعريف

الخط المستقيم باستخدام المعادلات البارمتريّة . عادة نحدد الخط المستقيم في الفراغ بنقطتين تقع عليه أو

بواسطة نقطة و متجه يوازي هذا الخط . (معرفة المتجهات vectors أذهب للفصل القادم) .



كما ذكرت الخط المستقيم في الفراغ يمثل بنقطتين تقع عليه (الحمراء و السوداء) .

النقطة الحمراء $P_1(X_1, Y_1, Z_1)$ و الأخرى $P_2(X_2, Y_2, Z_2)$.

لتحديد اتجاه الخط في الفراغ نقوم بفرض متجه وحدة Unit Vector و نسميه V و هذا المتجه

يساوي $\langle X_2 - X_1, Y_2 - Y_1, Z_2 - Z_1 \rangle$ و هو ما يناظر الميل كما سبق و له هذه المركبات

$\langle a, b, c \rangle$.

إذن المعادلة البارامترية للخط المستقيم في الفراغ هي :

$$P(t) = P_0 + V * t$$

حيث P_0 هي النقطة الحمراء كما في الشكل أو النقطة الأولى P_1 و المتجه هو عبارة عن فرق

الإحداثيات بين النقطتين P_1 ، P_2 و t هو بارامتر قيمته تتراوح في جميع الأعداد الحقيقية .

مجموعة الأعداد الحقيقية هي اتحاد مجموعة الأعداد الصحيحة الموجبة و السالبة بما فيهم المجموعة

الخالية (الصفر) .

نلاحظ أن البارامتر t قيمته غير محدده فما رأيك لو قمنا بحذفه ؟ ..

المعادلة التماثلية للخط المستقيم في الفراغ

فلنفرض أن متجه الوحدة $V(a, b, c)$ و هو كما ذكرت كالميل عندما نتعامل في بعدين فمن الممكن

اختزال المعادلة البارامترية إلى هذا الشكل :

$$\frac{(x - x_0)}{a} = \frac{(y - y_0)}{b} = \frac{(z - z_0)}{c}$$

a

b

c

و هذه هي المعادلة التماثلية للخط المستقيم حيث a, b, c هم مركبات متجه الوحدة أي (x, y, z) لهذا المتجه .

مثال : أوجد المعادلة التماثلية و البارمتريّة حيث يقع علي هذا الخط نقطتين الأولى $P_0(1, 2, 3)$ و الأخرى $P_1(5, 6, 7)$.

الحل :

$$(P_0 - P_1) = v$$

$$\langle 4, 4, 4 \rangle = \langle 5-1, 6-2, 7-3 \rangle =$$

المتجه = متجه الوحدة

مقدار (معياره)

$$\langle 4, 4, 4 \rangle =$$

$$\sqrt{4^2 + 4^2 + 4^2}$$

$$\langle 0.57, 0.57, 0.57 \rangle =$$

$$a = 0.57, b = 0.57, c = 0.57$$

إذن المعادلة التماثلية هي

$$x-1 = y-2 = z-3$$

$$.57 \quad .57 \quad .57$$

و لو قمنا بضرب المعادلة في 57. لإلغاء المقام سينتج أن

$$x-1 = y-2 \quad \text{و} \quad x-1 = z-3$$

و هما معادلتين الخط المستقيم في الثلاثة أبعاد .

و المعادلة البارمتريّة هي :

من المعلوم أن

$$x = x_0 + t \cdot a, \quad y = y_0 + t \cdot b, \quad z = z_0 + t \cdot c$$

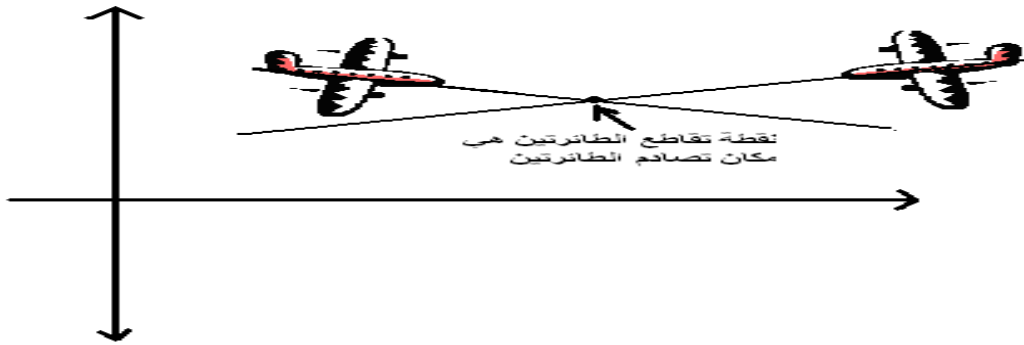
و بالتعويض في هذه المعادلات نجد أن

$$x = 1 + .57 \cdot t, \quad y = 2 + .57 \cdot t, \quad z = 3 + .57 \cdot t$$

و هذه هي المعادلات البارامترية للخط المستقيم (نلاحظ وجود t و يمكن حذف هذا البارامتر بوجود معطي آخر في المسألة أو المشكلة مثلاً إذا كان هذا الخط يقطع مستوي ما فمن هذا المعطي نستطيع تعيين نقطة التقاطع و لتعيينها يجب تعيين البارامتر t .

تطبيقات علي الخط المستقيم في برمجة الألعاب
هناك تطبيقات عديدة يمكن استنتاجها باستخدام معادلات الخط المستقيم و من أهم هذه التطبيقات كشف التصادم collision بين كائنات اللعبة .

فمثلاً فلنفرض أن هناك طائرتين تسلك مسار خط مستقيم كم موضح في الشكل فكيف نعرف أن الطائرتين اصطدما؟! .



و كما نلاحظ من الشكل أننا مثلنا مسار الطائرتين بخطين مستقيمين (مُمثل في بعدين) و من الممكن مد هذا النموذج إلي البعد الثالث .

فلنفرض أننا نريد أن نكشف إذا كانت الطائرتين في لعبتنا تصادما؟! ...

أي نريد أن نجلب نقطة تقاطع الخطين و كما تعلم فإن النقطة عبارة عن إحداثين أو ثلاثة علي ... و عند جلب هذه النقطة ستكون هي "موقع" تصادم الطائرتين أو الحالة التي فيها ستصطدم الطائرتان .
مثال:

فلنفرض أن طائرة ممثلة بمعادلة الخط المستقيم $x + 3y = 2$ و الأخرى تسير علي خط مستقيم (أو مسار trajectory) معادلته $4x + 2y = 3$ نريد أن نعلم عند أي نقطة سيتقاطع مسار الطائرتين؟! (لاحظ انه نستفهم عن مكان و ليس زمان أي متي يتقاطع المسارين لأننا لم ندمج الزمن في المعادلات و هذه ملحوظة مهمة جداً) .

من معني المسألة أننا نريد نقطة التقاطع المستقيمين و نستطيع تحديد هذه النقطة عن طريق حل معادلتين عن طريق الطرق المعروفة لذلك و منها

١. التعويض ٢. معكوس المصفوفات ٣. الحذف بجاوس ٤. رتبة المصفوفة ٥. طريقة كرامر.
هذه أهم طرق حل المعادلات الشهيرة و سوف نستخدم الطريقة الأولى (التعويض) للسهولة و هذه
الطريقة تتلخص في جعل متغير في طرف واحد ثم نعوض به في المعادلة الأخرى للحصول علي
المتغير الآخر .

$$X-3Y=2 \dots\dots\dots(1)$$

$$4X+2Y = 14 \dots\dots\dots(2)$$

$$x = 3Y + 2$$

و من هنا نعوض في المعادلة الأخرى

$$4(3Y+2) + 2 = 14$$

$$12Y + 8 + 2 = 14$$

$$12Y = -10 + 14$$

$$12Y = 4$$

و بعد ذلك نعوض في المعادلة (١) أو (٢) $Y = 1/3$

$$X-3Y = 2$$

$$X-1= 2$$

$$X =3$$

إذن نقطة التقاطع هي

$$(X , Y) = (3 , 1/3)$$

أي المساران سيتقاطعان عن هذه النقطة .

و لكن يا أحمد كيف أستخدم هذه الفكرة في برامجي ؟! لقد رأيت فقط مسألة رياضية و لم أر مسألة
برمجية ...

بالرجوع لتعريف الخط المستقيم تستطيع أن تستنتج ميل الخط المستقيم و الجزء المقطوع من المحور
الصادي .. الخ و من الممكن تحديد النقط التي سارت من و إلي النقطة الأخرى .. و يبقي لك حل
المعادلتين .

لتسهيل عملية الحل سوف نختزل معادلة الخط المستقيم هذه التي تعرفت عليه من قبل $y-y_1 = m(x-x_1)$

(x_1 إلي معادلة في مجهول واحد لتسهيل حل المسألة برمجياً

و بعد عمليات الاختزال سنجد أن

$$y = m1(x - x1) + y1$$

$$y = m2(x - x2) + y2$$

و بالتالي المتغير x سوف يساوي $x = (m1x1 - m2x2 + y2 - y1) / (m1 - m2)$

و هذا ما نحتاجه لحل المسألة .

و لحل المسألة برمجياً إليك هذا الكود

```
float *lineIntersect(float *L1Point, float L1Slope,
```

```
float *L2Point, float L2Slope)
```

```
{
```

```
// مصفوفة تحمل ثلاث عناصر تحمل قيم نقطة التقاطع
```

```
// حيث المتغير الأول من المصفوفة يساوي المتغير السيني
```

```
float temp[2] = {0, 0};
```

```
// هنا نحل المعادلة كما ذكرنا فيما سبق
```

```
temp[0] = (L1Slope * L1Point[0] - L2Slope * L2Point[0] +
```

```
L2Point[1] - L1Point[1]) / (L1Slope - L2Slope);
```

```
// بعد ما حصلنا علي قيمة س في السطر السابق نستخدم هذه القيمة في إيجاد القيمة ص
```

```
temp[1] = L1Slope(temp[0] - L1Point[0]) + L1Point[1];
```

هنا نهاية الدالة التي سترجع قيمة المصفوفة و التي تحتوي عناصرها علي متغيرات نقطة التقاطع //

أعط لنا قيمة المصفوفة ألان ! //

```
return temp;
```

```
}
```

المؤشر L1Point يمثل عنوان النقطة للخط المستقيم الأول

المؤشر L2Point يمثل عنوان النقطة للخط المستقيم الثاني

المؤشر L1Slope يمثل عنوان ميل خط المستقيم الأول

المؤشر L2Slope يمثل عنوان ميل خط المستقيم الثاني

المصفوفة temp تحمل متغيرات النقطة المطلوب إيجادها .

لاحظ أننا استخدمنا المصفوفات هنا لبساطة المسألة البرمجية و لم نستخدم هياكل البيانات الأخرى مثل

البيانات structures و القوائم المترابطة linked lists .. الخ .

الدائرة و الكرة

Sphere , Circle

الدائرة و الكرة من أهم الأشكال الهندسية في برمجة الألعاب و الرسومات و خصوصاً عند صناعة

المحركات .

ربما تريد أن ترسم دائرة مثلاً أو تريد أن تجعل كائن في لعبتك يدور في مسار دائري .. الخ من

الاستخدامات سوف تواجهها عند برمجة الألعاب .

دعنا أن نوضح معادلة الدائرة ; لأنه عن التعبير عن أي شكل هندسي تعتبر معادلته هي الخاصية

الهامة له ; لذا يجب الاهتمام بهم ..

و الدائرة هي عبارة عن مجموعة نقط تبعد مسافة تسمى نصف القطر من نقطة ثابتة Radius تسمى

المركز Center و هذا تعريفها في المستوي و لكن في الفراغ نفس التعريف يستخدم و لكن تسمى كرة

مثل كرة التنس .

و من نصف قطر الدائرة و مركزها نستطيع أن نمثل معادلة للدائرة و هي شبيها لنظرية فيثاغورث جداً

حيث باختصار النظرية تقول (أن مربع الوتر يساوي مربع الضلع المجاور و الضلع المقابل لمثلث قائم

الزاوية right triangle) .

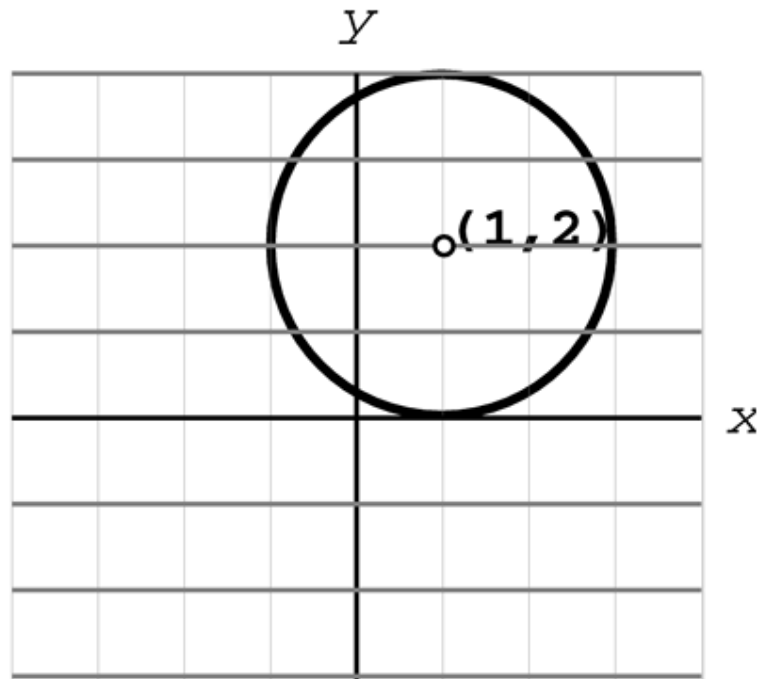
معادلة الدائرة

$$(x-h)^2 + (y-k)^2 = r^2$$

حيث :

النقطة (h,k) هي مركز الدائرة .

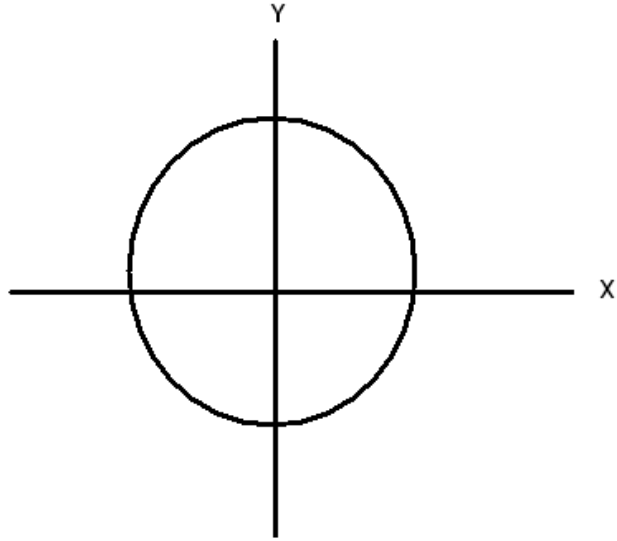
r نصف قطر الدائرة .



صورة تمثل رسم دائرة مركزها النقطة $(1, 2)$.

و معادلة الدائرة التي مركزها نقطة الأصل $(0, 0)$ هي

$$x^2 + y^2 = r^2$$



صورة توضح رسم دائرة و لكن " تقع " (مركزها) نقطة الأصل .
 إذا قمت بتدوير الدائرة حول محورها سوف تشاهد كرة و معادلة الكرة مثل معادلة الدائرة باستثناء انك
 أضفت بعد إضافي و هو البعد الثالث (العمق) Z .
 أي عندما تريد عمل لعبة ثنائية الأبعاد الدائرة والخط المستقيم .. الخ من الأشكال و لكن إذا كنت تريد
 أن تعمل لعبة ثلاثية الأبعاد فأستخدم الكرة و الخط المستقيم الواقع في الفراغ .
 معادلة الكرة لها مركز

$$(x-h)^2 + (y-k)^2 + (z-l)^2 = r^2$$

حيث النقطة الواقعة في الفراغ (h,k,l) و التي تحدد مركز الكرة هي مركز الكرة.
 R نصف قطر الكرة .
 معادلة الكرة التي مركزها نقطة الأصل

$$x^2 + y^2 + z^2 = r^2$$

نلاحظ إضافة متغير جديد z و هو البعد الثالث الذي يمثل الفراغ .

نظرة إلي التصادم في الألعاب
 لنفرض أن لدينا قطارين في لعب ما و نريد أن نكشف هل سيصطدما أم لا سوف نقرب المسألة بتمثيل
 القطارين بدائريتين تحيط بشكل القطار و هذا لتبسيط عملية التصادم لان في برمجة الألعاب من الصعب

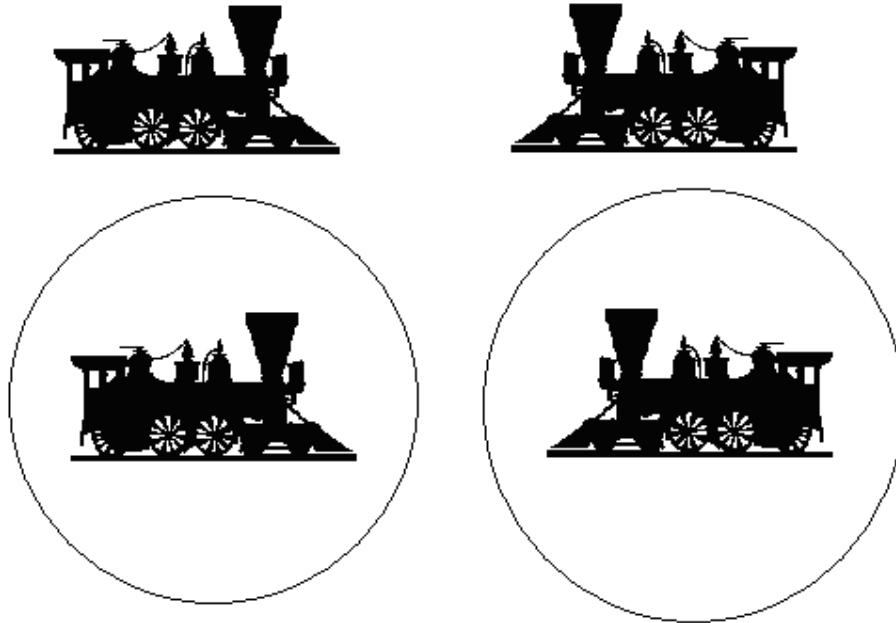
كشف تصادم كل جزء ما في الكائن و لكن نحن لا نقول أنها عملية لم نفعل من قبل في الألعاب فمن اشهر الألعاب لعبة فتوات الشوارع street fighters التي استخدمت أساليب معقدة في التصادم فعلي سبيل المثال عندما كانت شخصية ما تضرب عدو في الواجهة يحدث تصادم و أيضاً هناك كشف لتصادم الرجل مع البطن .. الخ من الأساليب ..

إذن الآن لنلخص هذا الكلام

١. هناك أنواع من التصادمات المعقدة و التي نحاول أن نحاكيها في ألعابنا مثل ألعاب الكاراتيه و التي يجب الكشف عن كل جزء في منطقة الجسم .
٢. هناك أنواع من التصادمات البسيطة مثل تصادم طلقة من مدفع إلي طائرة و هكذا..

تطبيقات الدائرة و الكرة في التصادمات

لتبسيط عملية التصادم في الألعاب فأنا غالباً نستخدم الأشكال الهندسية مثل الدائرة و المستطيل أو الكرة في عملية تسهيل الكشف عن التصادم .

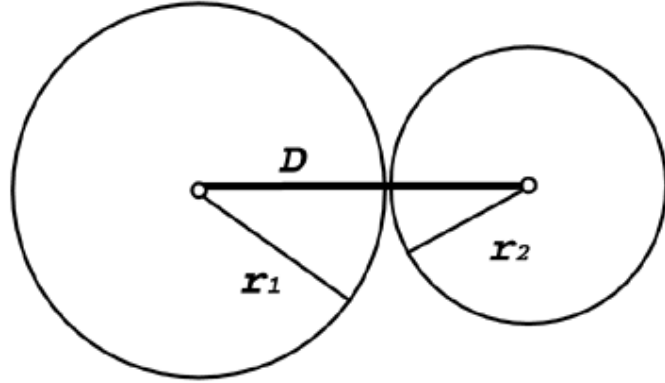


فلنفرض قطارين سائرين في لعبة ما و نريد أن نكشف تصادمهم معاً ..

للسهولة سوف نرسم دائرتين تحيط بالقطار (بالطبع الدائرة لن ترسم في اللعبة فهي دائرة وهمية) و هذه الطريقة تسمى Bounding Circle / Sphere الدائرة المحيطة بالمجسم أو المغلقة ...

قمنا بهذه العملية لان هذا تصادم بسيط لا يحتاج إلي تفاصيل كثيرة مثل الألعاب القتالية (سوف أشرح فكرتها بعد قليل) .

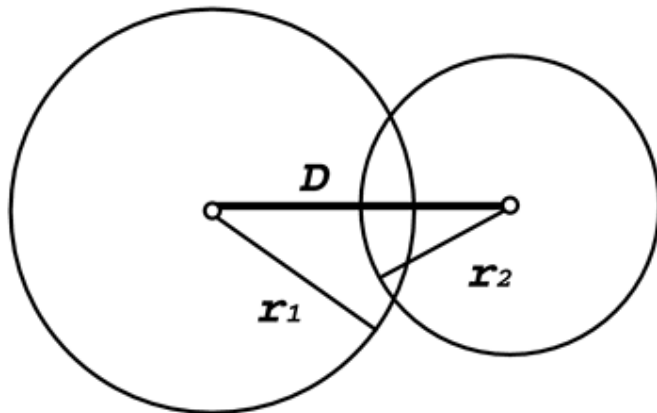
سوف نستخدم بعد الحسابات الرياضية لكشف التصادم ..



هذه الدائرتان موضحا في الشكل

نلاحظ أن هناك مسافة بين الدائرتين (الكائنين) ولقد سميناها D و في الحالة الموضحة في الرسم الدائرتين (الكائنين) لم يصدما و نلاحظ أن المسافة التي بينهما D تساوي مجموع نصفي قطري الدائرتين $R1 + R2$ و هذا يعني أن شرط تصادم الدائرتين أو تقاطعهما هو أن تكون المسافة بين مركزي الدائرتين d أقل من مجموع نصفي القطرين $R1 + R2$.

هذه القاعدة عاماً علي أي كائن محيط بدائرة .



و الصورة توضح ذلك

نعم هذه فكرة رائعة و ذكية **J** ! و طريقة رياضية سليمة 100% .

و لكن كيف نحسب المسافة بين مركزي الدائرة ؟! هذا بقانون المسافة (قانون فيثاغورث) .

إذن القاعدة هي :

إذا كان هناك دائرتين معرفين بالمعادلات فكما علمت أن الدائرة تُحدد بمركز و نصف قطر

فسنفرض أن الدائرة الأولى معادلتها : $(x-h_1)^2 + (y-k_1)^2 = r_1^2$

الدائرة الثانية معادلتها: $(x-h_2)^2 + (y-k_2)^2 = r_2^2$

إذن شرط حدوث التصادم هو أن يكون

$$\sqrt{(h_2 - h_1)^2 + (k_2 - k_1)^2} \leq (r_1 + r_2) \quad \text{ـ}$$

حيث الطرف الأيسر هو المسافة

و الطرف الأيمن هو مجموع أنصاف الأقطار .

العملية بسيطة أليس كذلك ؟!

ملحوظة: لا يجب أن يكون الكائنين في نفس الحجم أي الدائرتين في نفس الشكل يمكن تطبيق هذه

القاعدة علي أي نوع من الأشكال .

مسألة رياضية

فلنفرض أنك تقوم بصنع لعبة سباق سيارات و لكن في بعدين 2D و جاء ألان كشف التصادمات بين

السيارات و كانت سيارتان تسيران في السباق و كانت السيارة الأولى محيطة بدائرة معادلتها $(x-50)^2$

$+ (y-20)^2 = 900$ و السيارة الأخرى معادلتها

$(x+10)^2 + (y-10)^2 = 400$ هل حدث تصادم بينهما ؟!

أتبع هذه الخوارزم

أولاً نحدد مركز الدائرة الأولى و هو (50,20) و نصف قطرها 30

الدائرة الأخرى مركزها (10,-10) و نصف قطرها 40

ثانياً نحسب قيمة المسافة بين السيارتين (الدائرتين)

عن طريق القانون الذي ذكرته و بعض التعويض بالمعطيات الموجودة فوق

$$D = \sqrt{(h_2 - h_1)^2 + (k_2 - k_1)^2}$$

$$D = \sqrt{(-10 - 50)^2 + (10 - 20)^2}$$

$$D = \sqrt{(-60)^2 + (-10)^2}$$

$$D = \sqrt{3600 + 100}$$

$$D = \sqrt{3700}$$

$$D \approx 60.83$$

نجد أن المسافة هي 60.83 الوحدة .

ثالثاً نكشف إذا كانت هذه المسافة أقل من أو تساوي مجموع نصفي قطري الدائرتين

نلاحظ أن مجموع القطرين

$$R_1 + R_2 = 50$$

و المسافة بين الدائرتين D تساوي 60.83 تقريباً

و نلاحظ أن الشرط تحقق (مجموع أنصاف الأقطار أقل من المسافة بين الدائرتين) إذن نستنتج أن السيارتين تصادما ! .

الآن تستطيع مد هذا القانون للكرة (البعد الثالث) و الألعاب الثلاثية الأبعاد و القاعدة هي نفس القاعدة التي ذكرت مع زيادة البعد الجديد .

القانون :

$$(x-h_1)^2 + (y-k_1)^2 + (z-l_1)^2 = r_1^2 \text{ نفرض كرة معادلتها}$$

$$(x-h_2)^2 + (y-k_2)^2 + (z-l_2)^2 = r_2^2 \text{ و كرة أخرى ,}$$

شرط التقاطع (التصادم)

$$(h_2 - h_1)^2 + (k_2 - k_1)^2 + (l_2 - l_1)^2 \leq (r_1 + r_2)^2$$

نقطة هامة :

يُفضل مبرمجون الألعاب عدم استخدام الجذور التربيعية Square Roots و الدوال المثلثية مثل الجا و الجتا Sine,Cosine لأنها حقاً تهلك معالج الحاسب حيث انه لتنفيذ عملية أخذ الجذر رقم يستهلك المعالج عمليات عديدة لتنفيذ هذه العملية و لتفادي هذه المشكلة هناك عدة طرق الأفضل هي استخدام لغة الاسبمبلي (التجميع) Assembly Language و لكن لماذا ؟! لان لغة التجميع هي اللغة الوحيدة الأقرب إلي لغة الآلة أي الأقرب للمعالج و بالتالي ستكون العملية أسرع بدلاً من كتابة الكود في أحد لغات البرمجة مثل السي ، السي ++ التي يتم تحويلها للمترجم لكود لغة اسيمبلي و من ثم اللغة الآلة و هذه العملية بطيئة!.

و لكن للتبسيط هنا و لعدم التعقيد سوف اختزل معادلة المسافة التي فوق من الجذر بتربيع الأطراف .

إذن القاعدة الجديدة :

$$(h_2-h_1)^2 + (k_2-k_1)^2 \leq (r_1+r_2)^2$$

نظرة برمجية للدائرة و الكرة

يمكن تمثيل الدائرة و الكرة برمجياً عن طريق هيكل بيانات يحدد خصائص الدائرة المهمة مثل نصف القطر و المركز .. مثل الكود هذا

بداية الهيكل // struct circle

```
{
    مصفوفة تحوي علي متغيرات مركز الدائرة // float center[2];
    متغير يحمل نصف قطر الدائرة // float radius;
};
```

struct sphere

```
{
    float center[3];
    float radius;
};
```

لنقم بكتابة دالة تكشف إذا كانت كرتان تصادما أو لا

bool IsTheyCollided(sphere &S1, sphere &S2)

```
{
```

```

return (pow(S2.center[0] - S1.center[0], 2) +
        pow(S2.center[1] - S1.center[1], 2) +
        pow(S2.center[2] - S1.center[2], 2)) <

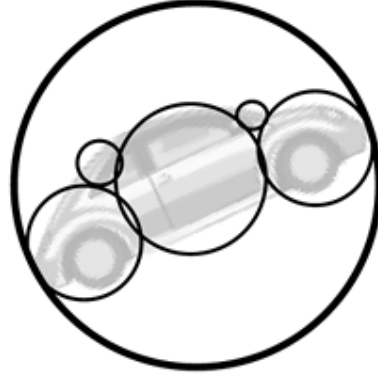
        pow(S1.radius + S2.radius, 2));
}

```

الكود سهل فلقد أستخدمنا الدالة pow (فهي تقوم بتربيع بارامتراتهما) الموجودة ضمن مكتبات السي++
. math.h

التصادمات المركبة المعقدة

لقد تحدثنا عن بعض المسائل البرمجية الموجودة في برمجة الألعاب و التي تخص مشكلة التصادمات و لكن هذه المشاكل تُعتبر بسيطة مثل تصادم سيارتين ، قطارين .. الخ من الأمثلة ماذا عن كشف التصادمات و لكن في الألعاب القتالية التي تحتوي علي ضرب في الرأس و البطن و جميع الأجزاء من الجسم كيف نقوم بحل هذه المشكلة؟! . من ابسط الطرق لحل هذه المشكلة هي اعتبار كل كائن و ليكن شخصية قتالية في لعبة فتوات الشوارع أنه عبارة عن دائرة و إن لم تكن الدائرة تحيط بالأجزاء نستطيع أن نستخدم شكل آخر مثل المستطيل لحل المسألة (سوف نتطرق لهذا الشكل بعد الفاصل J) . هذا هو الجزء الفني و هو اعتبار مثلاً قدم الشخصية دائرة و راس الشخصية دائرة و يد الشخصية دائرة و لكن في الحقيقة مثل هذه النوعية من المشكلة لا نحبذ استخدام الدائرة في تمثيل أعضاء الشخصية القتالية و من الأفضل أن نمثلها كمستطيل و كمثال لاستخدام دوائر متعددة لتحقيق تصادم دقيق أنظر لهذه الرسمة و التي تحتوي علي دوائر لكل جزء في السيارة فمثلاً نريد أن نكشف تصادم (إطارات) السيارة مع الحائط .. الخ من الكشوفات.



المستويات

Planes

تكلمنا في الأقسام السابقة عن النقطة و المستقيم في المستوي و الفراغ و سنتكلم الآن عن المستويات .
ما هي المستويات و خصائصها ؟

المستويات هي ابسط السطوح في الثلاثة أبعاد (الفراغ) و هو ذلك السطح الذي ينطبق عليه الخط المستقيم في جميع أوضاعه (هناك أوضاع للمستوي و لكنه خارج موضوع الكتاب لعدم أهميتها في برمجة الألعاب) في و المستوي ممتد في جميع الجهات إلي ما لا نهاية و يمكن أن يتعين عن طريق أربع طرق .

١ . ثلاث نقط ليست علي استقامة واحدة (مهمة جداً)

٢ . مستقيم و نقطة ليست علي استقامته

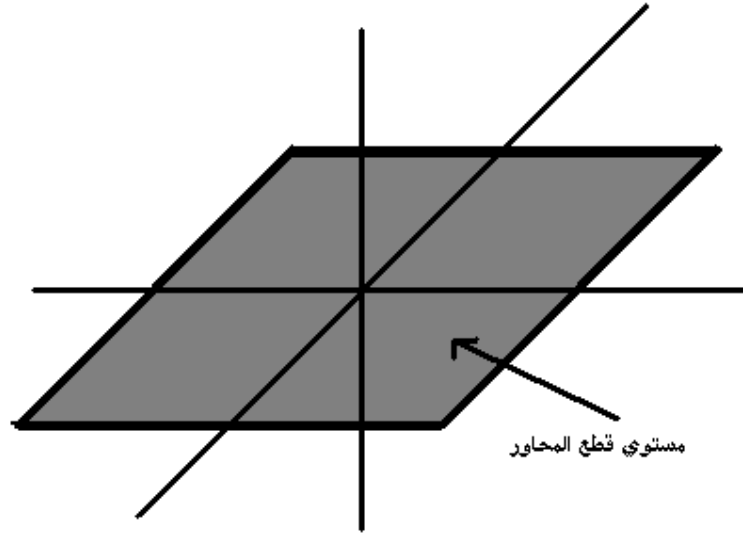
٣ . أي مستقيمان متقاطعان (في الفراغ) يكونان مستوي

٤ . أي مستقيمان متوازيان (في الفراغ) يكونان مستوي

أول نقطة في طرق تعيين المستوي ذكرت أنها مهمة و سنعلم لماذا عندما نتحدث عن المثلثات . حيث يتضح من التعريف أنه يمكن أن نعتبر المثلث Triangle في الفراغ هو عبارة عن مستوي و هذه الجملة مهمة جداً في صناعة محركات ثلاثية الأبعاد 3D Engines حيث كما سنعلم أن المجسمات

Meshes

اعتبرناها مكونة من مثلث لان كما قلت أن أي ثلاث نقط ليست علي استقامة واحدة تكون و مثلث و هناك أسباب عديدة لاختيار المثلث ..



صورة توضح شكل مستوي تحدد بتقاطع المستقيمات في الفراغ.
نلاحظ أيضاً أن المستوي يقسم الفراغ إلي نصفين و هي ملحوظة مهمة جداً لتصميم محركات ثلاثية الأبعاد ...

لكي نقوم بتمثيل المستوي جبرياً (أي بكتابة معادلاته) يجب المعرفة بجبر المتجهات و هو مختص بالفصل القادم و ما سأفعله هنا هو كتابة المعادلات و شرحها باعتبار أنك أخذت فكره عنه بعد قراءة الكتاب بأكمله .

تمثيل المستوي

فلنفرض أن هناك نقطة ثلاثية الأبعاد (واقعة في فراغ) P_o معلومة و كان هناك متجه $\langle n \rangle$ غير صفري معلوم فان فئة مجموعة النقط P بحيث أن المتجه $\langle PoP \rangle$ عمودي علي المتجه $\langle n \rangle$ تعرف علي إنها مستوي يمر بالنقطة P_o عمودياً علي المتجه $\langle n \rangle$.

فإذا فرضنا أن $P_o (x_o, y_o, z_o)$ و $P(x, y, z)$ و المتجه $\langle n \rangle = \langle a, b, c \rangle$ فإن

$$\langle PoP \rangle = \langle x - x_o, y - y_o, z - z_o \rangle$$

و من شرط تعامد متجهين نجد أن

$$\langle n \rangle \cdot \langle PoP \rangle = 0$$

أي أن

$$A (x - x_o) + b (y - y_o) + c (z - z_o) = 0$$

و هذه هي معادلة المستوي في الفراغ .