

الدرس الخامس عشر

أهلاً بكم في الدرس الخامس عشر من سلسلة دروس تعلم الـ Xna، في هذا الدرس سوف أقوم بالشرح عن مصفوفات التحويل "Transformation" ثنائية الأبعاد.

في هذا الدرس، سوف نرى كيف بإمكاننا استخدام الدالة `TexturesCollide`، التي قمنا بإنشاءها في الدرس السابق. من تعريف الدالة، نستطيع ان نستدل أنه يلزمنا مصفوفة ألوان ثنائية الأبعاد إضافة إلى مصفوفة التحويل للصورتين.

دعنا نبدأ بمصفوفة الألوان ثنائية الأبعاد. بما أننا سوف نستخدم الدالة `TexturesCollide` بشكل متكرر، سوف نقوم بإستخراج المصفوفة الخاصة بالصورة لمرة واحدة بإستخدام الدالة `TextureTo2DArray`، و نقوم بتخزينها في متغير. إبدأ بإضافة المتغيرات التالي إلى أعلى الكود:

```
Color[,] rocketColorArray;  
Color[,] foregroundColorArray;  
Color[,] carriageColorArray;  
Color[,] cannonColorArray;
```

لاحظ أن هذه هي الصور فقط التي نريد أن نفحص تصادمها، الصاروخ التضاريس و اللاعبين. قم بتحميل 3 منهم في نهاية الدالة `LoadContent`:

```
rocketColorArray = TextureTo2DArray(rocketTexture);  
carriageColorArray = TextureTo2DArray(carriageTexture);  
cannonColorArray = TextureTo2DArray(cannonTexture);
```

مصفوفة الألوان الخاصة بالتضاريس (الأرضية الأمامية) سوف نحتاج لأن نستخرجها في كل مره يتم إستدعاء الدالة `CreateForeground`، إذن قم بإضافة الكود التالي في نهاية تلك الدالة:

```
foregroundColorArray = TextureTo2DArray(foregroundTexture);
```

الآن لدينا مصفوفات الألوان الخاصة بالخامات جاهزه، هناك شيء واحد إضافي سوف نحتاجه: و هي مصفوفات التحويل لكل من تلك الصور. هذه المصفوفات تقوم بوصف كيف تم تحويل هذه الصور (تحريكها، تدويرها، تحجيمها) قبل رسمها على الشاشة. سوف نقوم بإنشاء مصفوفات التحويل للصور الأربعة، بداية بالأسهل: مصفوفة التحويل الخاصة بصورة التضاريس.

بالنسبة لصورة التضاريس، هذا سهل إلى حد ما. تذكر أننا قمنا بإنشاء هذه الخامة بأنفسنا، إذن لها نفس حجم الشاشة، بمعنى أن صورة التضاريس سوف يتم رسمها على الشاشة كما هي بالضبط. هذا يعني أننا نحتاج إلى مصفوفة تحويل لا تغير شيئاً على الصورة.

هذا يعني أننا نحتاج إلى تعريف مصفوفة محايدة لكي تمثل مصفوفة التحويل لصورة التضاريس. المصفوفة المحايدة تهتم بأن تبقى الصورة كما هي تماماً. يمكنك مقارنتها مع العنصر المحايد لعملية الجمع أي الصفر 0: رقم + 0 = نفس الرقم. بالنسبة للضرب العنصر المحايد هو الرقم 1: رقم * 1 = نفس الرقم. أما بالنسبة للمصفوفات : اي صورة * `Matrix.Identity` = نفس الصورة.

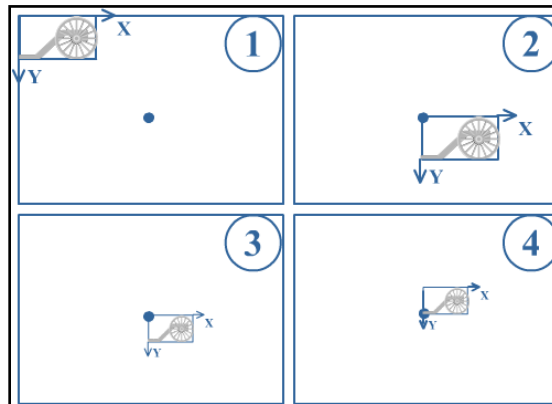
إذن هذا السطر يقوم بإنشاء مصفوفة لخامة التضاريس (لا تكتبه الآن):

```
Matrix foregroundMat = Matrix.Identity;
```

بعد ذلك نحتاج إلى المصفوفة الخاصة بحامل المدفع. قبل أن يتم رسمه على الشاشة، يتم تحريكه إلى الموقع الصحيح، و يتم تصغيره و يتم تحريكه مره أخرى إلى موقعه الأصلي. لكل من هذه الحركات تحويل "Transformation" منفصل، باستطاعتنا إنشاء المصفوفة له. مصفوفة التحويل العامة ما هي إلا ناتج ضرب المصفوفات الخاصة بكل من هذه التحويلات المنفصلة. هناك قاعدة مهمة جدا يجب أن نتذكرها: أن ترتيب ضرب المصفوفات مهم. في ضرب المصفوفات يجب أن تترجم الرمز "*" إلى "بعد".

بعد الإشارة إلى ذلك دعنا ننتقل إلى التحويلات التي يقوم ال Xna بعملها قبل رسم حامل المدفع على الشاشة. مهم جدا: أن هذه العمليات تتم حسب الترتيب المحدد هنا بالضبط. هذه الخطوات موضحة في الصور في الأسفل.

- 1) إذا أردنا أن نرسم الصورة كما هي تماما (أي: باستخدام المصفوفة المحايدة)، سوف يتم رسمها بحجمها الأصلي في الركن العلوي الأيسر للشاشة.
- 2) أولا، حامل المدفع يتم تحريكه بحيث تكون نقطة الركن العلوي الأيسر في الموقع المحدد في الوسيط الثاني في دالة الرسم ال SpriteBatch.Draw.
- 3) بعدها يتم تصغير كل شيء. كما تلاحظ هذا يتضمن الصورة، إضافة إلى الإحداثي X و Y لها.
- 4) أخيرا، هذه الخطوه هي أكثر الخطوات "تحديا": يتم تحريك الصورة على محور Y، بما أننا قمنا بتحديد الموقع (0, carriageTexture.Height) كنقطة الأصل (المركز للصورة). من المهم: أنها قد تحركت على محور Y الخاص به، الذي تم تصغيره. إذا بدلا من أن تتحرك على 39 بكسل على الشاشة، سوف يتم تحريك حامل المدفع بشكل عمودي بقيمة $(0.4 * 39) = 16$ بكسل (بما أن $\text{carriageTexture.Height} = 39$ و $\text{playerScaling} = 0.4$).



لاحظ أن الوضع سوف يكون مختلفا في حال تم بتديل الخطوه الثانية بالخطوة الثالثة: اولا سوف يتم تحريك الصورة 39 بكسل بشكل عمودي، و بعدها سوف يتم تصغير الصورة. هذا ما سيؤدي إلى رسم حامل المدفع في الهواء!

دعنا الآن نعيد كتابة المصفوفات التالية:

- Matrix.Identity - 1
- Matrix.CreateTranslation(xPos,yPos,0) - 2
- Matrix.CreateScale(playerScaling) - 3
- Matrix.CreateTranslation(0,-carriage.Height,0) - 4

مصفوفة التحويل العامة يتم الحصول عليها من خلال ضرب كل تلك المصفوفات معا:

```
Matrix carriageMat = Matrix.CreateTranslation(0, -carriage.Height, 0) *
Matrix.CreateScale(playerScaling) * Matrix.CreateTranslation(xPos, yPos, 0) *
Matrix.Identity;
```

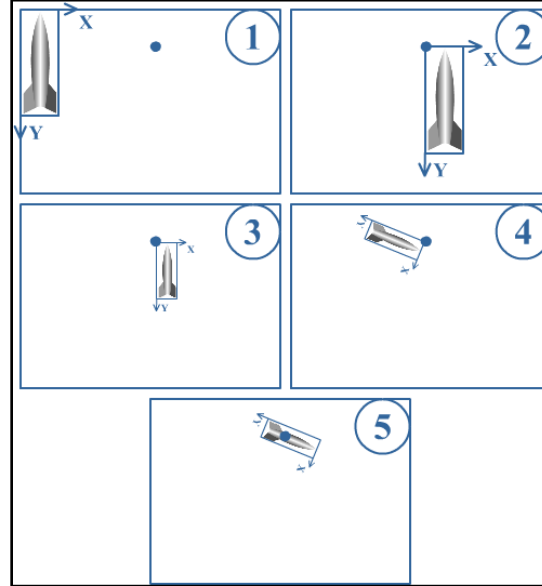
تذكر مناقشتنا حول المصفوفة المحايدة، بمعنى أنه يمكننا عدم وضعها في الكود لأنها لن تؤثر في النتيجة، إذن ينتج لدينا:

```
Matrix carriageMat = Matrix.CreateTranslation(0, -carriage.Height, 0) *  
Matrix.CreateScale(playerScaling) * Matrix.CreateTranslation(xPos, yPos, 0);
```

بعدها، هذا ينقلنا إلى المثال الأكثر شمولاً، قد حان الوقت لإنشاء مصفوفة التحويل الخاصة بالصاروخ دعنا نمر على التحويلات المختلفة التي تتم على الصاروخ، أثناء ذلك ألق نظرة على الصورة في الأسفل:

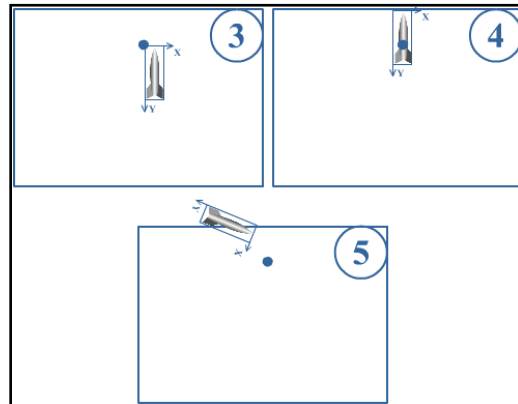
- 1) إذا أردنا رسم الصورة كما هي بالأصل (بمعنى: باستخدام المصفوفة المحايدة)، سوف يتم رسمها بحجمها الأصلي في الركن العلوي الأيسر للشاشة.
- 2) أولاً، يتم تحريك الصاروخ بحيث يصبح الركن الأيسر العلوي موجود على النقطة المحددة في الوسيط الثاني في دالة الرسم `SpriteBatch.Draw`.
- 3) بعدها، يتم تصغير كل شيء. كما ترى أن ذلك يشمل الصورة، بما يشمل محاور ال `X` و ال `Y` الخاصة بها.
- 4) بعدها، يتم تدوير كل شيء. مرة أخرى ذلك يشمل الصورة و محاور ال `X` و ال `Y` الخاصة بها.
- 5) أخيراً، يتم تحريك الصورة الناتجة 42 بكسل على محور `X` الخاص بالصورة، و 240 بكسل على محور `Y` الخاص بالصورة أيضاً. لنعرف كيف يتم التحريك بالنسبة للشاشة، دعنا نحدد أي من التحويلات للصورة تقوم بالتأثير على محور `X` و كذلك على محور `Y`. في الخطوة الثانية، تم تصغير المحاور 10 مرات (بما أن `rocketScaling = 0.1`). في الخطوة الثالثة تم تدوير هذه المحاور المصغرة بالزاوية `rocketAngle`. هذا يعني، أنه في حالة كانت الزاوية = 0، سوف يتم تحريك الصورة ب (4.2, 24) بكسل. في حالة الدوران، سوف يتم تحريك الصورة (42, 240) بكسل على محاورها، الذي تم تصغيرها و تدويرها. كمثال: في الخطوة الثانية، قمنا بتصغير صورة الصاروخ عشر مرات. دعنا نفترض أننا لم نقوم بعمل تدوير في الخطوة الثالثة. كنتيجة، سوف يتم تحريك الصورة (4.2, 24) بكسل.

بما أن هذه هي الحالة الأكثر شمولاً، سوف يكون بإستطاعتك إيجاد مصفوفة التحويل لأي صورة تقوم برسمها بإستخدام هذه القاعدة.



لاحظ أن ذلك يحقق النتيجة التي نريدها: مركز التدوير للصاروخ هو في الموقع الذي حددناه في الدالة `SpriteBatch.Draw`، و الصاروخ قد تم تدويره و تحجيمه كما نريد.

لقد سبق وذكرنا أن الترتيب في عمليات التحويل مهم. كمثال الصورة التالية توضح لنا ماذا سوف يحصل في حال تم التبديل بين الخطوتين 4 و 5 :



بما نظام الإحداثيات سوف يتم تحريكه أولاً، بعدها سوف يتم تدويره، سوف ينتهي في موقع آخر تماماً. هذا سوف يؤدي إلى رسم الصاروخ خارج الشاشة بشكل جزئي.

دعنا نعيد كتابة التحويلات بالطريقة الصحيحة، هذه المرة باستخدام المصفوفات. لاحظ أيضاً أننا تركنا المصفوفة المحايدة كما فعلنا في الخطوة الأولى:

- أولاً: `Matrix.CreateTranslation(rocketPosition.X,rocketPosition.Y,0)`
- بعدها: `Matrix.CreateScale(rocketScaling)`
- بعدها: `Matrix.CreateRotationZ(rocketAngle)`
- أخيراً: `Matrix.CreateTranslation(-42,-240,0)`

بما أننا نريد أن ندمج كل المصفوفات معاً، سوف نقوم بضربهم. تذكر أن "*" تعني "بعد":

```
Matrix rocketMat = Matrix.CreateTranslation(-42, -240, 0) *
Matrix.CreateRotationZ(rocketAngle) * Matrix.CreateScale(rocketScaling) *
Matrix.CreateTranslation(rocketPosition.X, rocketPosition.Y, 0);
```

هذا هو المثال الأشمل عن التحويلات التي قد تصادفها في لعبة ثنائية الأبعاد. كمثال: المصفوفات الخاصة بكل من خامسة التضاريس و صورة المدفع هما تبسيط للسطر السابق. يمكنك بناء على السطر `SpriteBatch.Draw` أن تقوم بإنشاء هذا السطر (الخاص بالمصفوفات). آخر مصفوفة إلى اليسار تحتوي على نقطة الأصل التي قمت بتحديددها في الدالة `SpriteBatch.Draw`, بعدها الزاوية, بعدها التحجيج يليها الموقع النهائي على الشاشة، الذي قمت بتحديدده في الوسيط الثاني في استدعاء دالة الرسم.

كمثال، دعنا نرى مدى سهولة إنشاء مصفوفة التحويل الخاصة بالمدفع، بناء على استدعاء الدالة `SpriteBatch.Draw`:

```
spriteBatch.Draw(cannon, new Vector2(xPos + 20, yPos - 10), null,
player.Color, player.Angle, cannonOrigin, playerScaling, SpriteEffects.None,
1);
```

و هذه هي المصفوفات الخاصة بها:

```
Matrix cannonMat = Matrix.CreateTranslation(cannonOrigin) *  
Matrix.CreateRotationZ(player.Angle) * Matrix.CreateScale(playerScaling) *  
Matrix.CreateTranslation(xPos + 20, yPos - 10, 0);
```

هذا هو كل شيء في هذا الدرس. في هذه اللحظة، لدينا مصفوفات الألوان الثنائية البعد للصور، إضافة إلى مصفوفات التحويل لهذه الصور، طبعا إضافة إلى الدالة CheckTextures. سوف نقوم بدمج كل شيء معا في الدرس القادم إن شاء الله.

للتذكير فقط :

```
Matrix foregroundMat = Matrix.Identity;  
  
Matrix carriageMat =  
    Matrix.CreateTranslation(0, -carriage.Height, 0) *  
    Matrix.CreateScale(playerScaling) *  
    Matrix.CreateTranslation(xPos, yPos, 0) *  
    Matrix.Identity;  
  
Matrix rocketMat =  
    Matrix.CreateTranslation(-42, -240, 0) *  
    Matrix.CreateRotationZ(rocketAngle) *  
    Matrix.CreateScale(rocketScaling) *  
    Matrix.CreateTranslation(rocketPosition.X, rocketPosition.Y, 0);  
  
Matrix cannonMat =  
    Matrix.CreateTranslation(cannonOrigin) *  
    Matrix.CreateRotationZ(player.Angle) *  
    Matrix.CreateScale(playerScaling) *  
    Matrix.CreateTranslation(xPos + 20, yPos - 10, 0);
```