MONOGAME 2.5D

Introducción

El primer objetivo es mostrar figuras primitivas en pantalla, que puedan rotar, trasladarse y escalarse en 3D, sin añadir perspectiva, ni luces, ni demás efectos 3D.

Para ello hay que seleccionar la cámara adecuada y las transformaciones correctas para cada elemento.

Cámara

La cámara es común para todos los elementos que se muestran en pantalla y es definida por dos matrices:

1. Hacia donde mira (View)
2. Como proyecta (Projection)

```
internal class Camera
{
    public Matrix View = Matrix.Identity;
    public Matrix Projection = Matrix.Identity;

    public Camera(Vector3 pos, Vector3 target, Vector3 up,
        float left, float top, float right, float bottom,
        float near = -10, float far = 10)
    {
        View = Matrix.CreateLookAt(pos, target, up);

        //2D View
        Projection = Matrix.CreateOrthographicOffCenter(left, right, bottom, top, near, far);
    }
}
```

View

Hacia donde mira la cámara se define por tres parámetros

1. Position, Indica la posición en el espacio donde esta la cámara.
2. Target, Indica hacia donde mira.
3. Up, Indica como esta girada respecto del plano en el que mira.

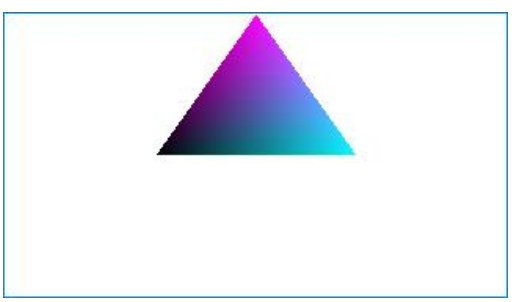
Teniendo esto en cuenta, supongamos el siguiente caso:

Caso 1

1. La cámara se encuentra en 1Z eso significa que esta hacia nosotros 1 unidad. Position = 0,0,1
2. Mira hacia el origen de coordenadas, Target = 0,0,0.
3. No esta girada, Up = 0,1,0

```
camera = new Camera(new Vector3(0, 0, 1), Vector3.Zero, Vector3.Up, -5, 5, 5, -5);
```

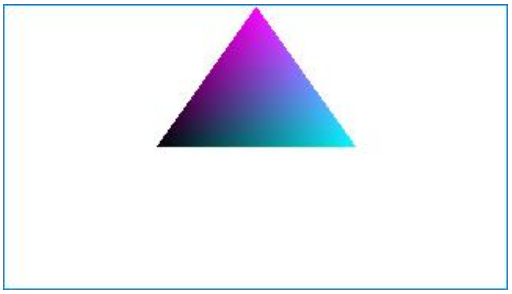
Teniendo en cuenta estos parámetros(sin tener en cuenta la proyección, ni el triangulo que aun hemos definido) tendríamos:



Caso 2

Cámara en 10Z:

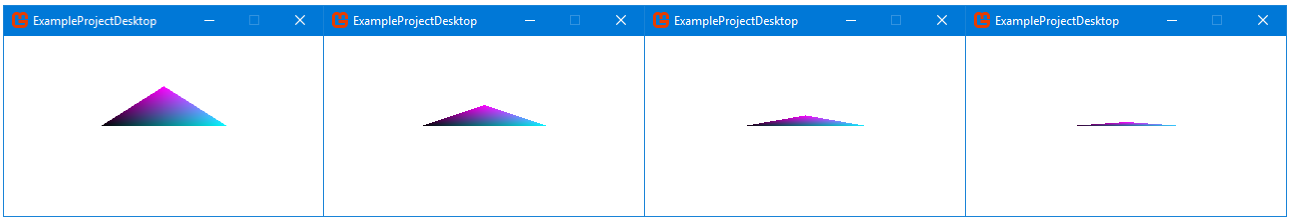
```
camera = new Camera(new Vector3(0, 0, 10), Vector3.Zero, Vector3.Up, -5, 5, 5, -5);
```



Ambos casos son iguales, porque no estamos usando perspectiva, es el propósito del 2.5D, queremos que aunque cambiemos la cámara de sitio:

Caso 3

En este caso vamos a cambiar position Y:




```
camera.View = Matrix.CreateLookAt(new Vector3(0, offset, 1), Vector3.Zero, Vector3.Up);
```

Los valores mostrados para offset son 2,4,6,10.

La rotación de la cámara es como inversamente exponencial, es decir, al principio se ve el triangulo tumbarse rápidamente y posteriormente va más despacio.

¿Por qué hasta 10?

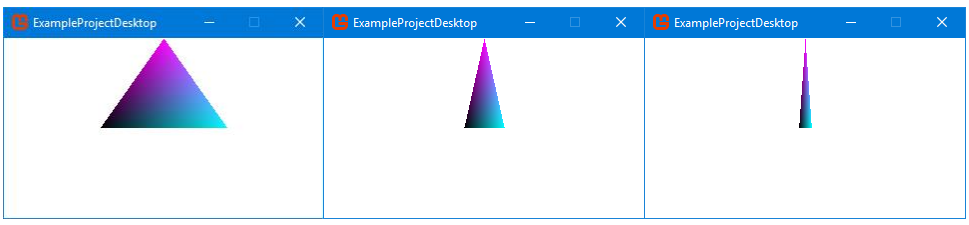
En la proyección que estamos utilizando, utilizamos un valor de plano lejano de 10, que interpreto como que la Y pasa a ser Z cuando llega ese valor.

Si utilizásemos de 0 a 50 ocurre cuando llega a este valor.

NOTA: Si utilizamos valores más pequeños, al cambiarlo, no se representan correctamente las figuras.

Caso 4

En este caso vamos a cambiar position X de la cámara:



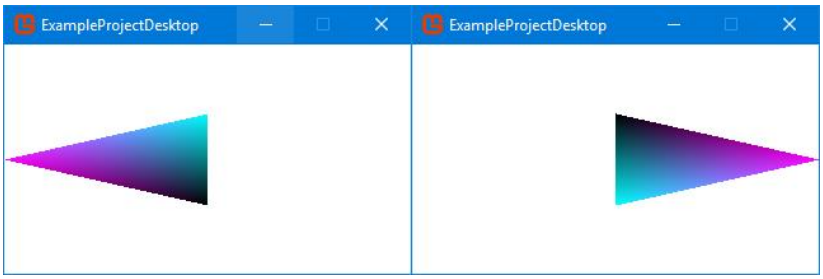
La cámara se desplaza en el eje X y siempre apunta al 0,0,0.

NOTA: Si utilizamos un position con un Z=0 no se dibuja el triangulo.

NOTA: Cambiamos el valor del camera target, En este caso el triangulo gira pero se desplaza un poco, ya que al fin y al cabo estamos a 1Z de distancia. Gráficamente no merece la pena mostrarlo.

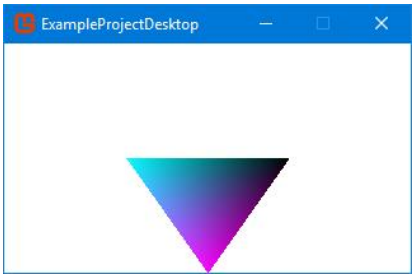
Caso 6

Ahora vamos a cambiar el último vector el CameraUpVector en el eje X sin sumar el Vector Up. Según sea X mayor o menor que cero, se produce un reflejo del triangulo.



Caso 7

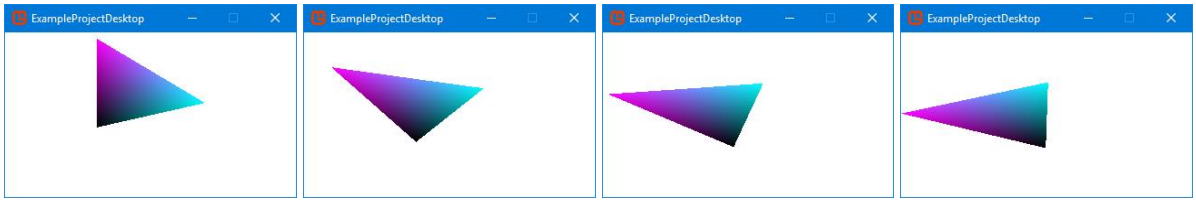
Ahora vamos a cambiar el último vector el CameraUpVector en el eje Y sin sumar el Vector Up. En este caso solo se invierte la imagen en el eje Y.



Caso 8

Ahora vamos a cambiar el último vector el CameraUpVector en el eje X sumándole el Vector Up.

```
camera.View = Matrix.CreateLookAt(new Vector3(0,0,1),Vector3.Zero,Vector3.Up+new Vector3(offset, 0,0));
```



Vemos como se va deformando y girando.

NOTA: Cambiar el eje Z del CameraUpVector no produce ningún resultado significante.

Conclusión

De momento los mejores parámetros de la cámara son los siguientes:

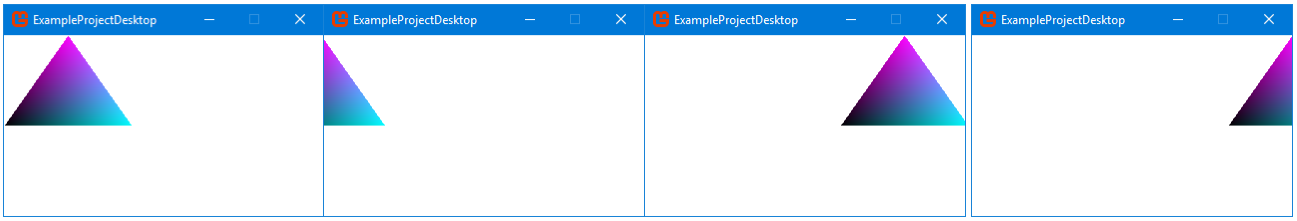
```
camera = new Camera(new Vector3(0, 0, 1), Vector3.Zero, Vector3.Up, ...);
```

Translación en el eje X

Vamos a controlar el movimiento de la cámara, teóricamente será cambiar a la vez en un eje o varios tanto la posición como el target.

```
camera.View = Matrix.CreateLookAt(new Vector3(offset, 0, 1), new Vector3(offset,0,0), Vector3.Up);
```

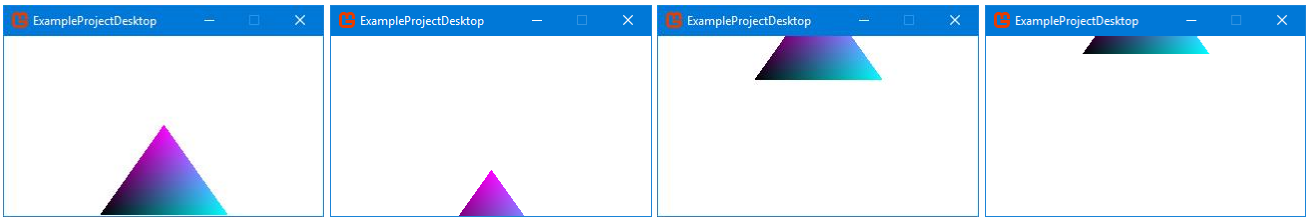
Para valores de offset en X: 3,5,-3,-5



Traslación en el eje Y

Para valores de offset en el eje Y: 5,7, -2.5, 4

```
camera.View = Matrix.CreateLookAt(new Vector3(0, offset, 1), new Vector3(0, offset, 0), Vector3.Up);
```



NOTA: Cambiarlo en eje Z no lleva ningún resultado por la proyección que estamos utilizando.

Conclusión

Al crear un entorno 2.5D podemos trasladar la cámara en el eje X y en el Y, aunque no podemos hacerlo en el Z.

Como veremos más adelante en el eje Z podemos acercar y alejar los elementos que estén en el.