MONOGAME 2.5D

Introduction

The first objective is show primitive figures in the screen, that can rotate, translate and scale in 3D, without adding perspective, lights neither 3D effects.

To achieve this we have to choose an adequate camera and correct transformations per each element.

Camera

The camera is a common element for all the elements shown in the screen and is define with two matrixes:

1. Towards is pointing (View)
2. How it projects (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);

        //2.5D View
        Projection = Matrix.CreateOrthographicOffCenter(left, right, bottom, top, near, far);
    }
}
```

View

Towards the camera points is defined by three parameters

1. Position, Indicates the position in the space of the camera.
2. Target, Towards it looks.
3. Up, How the camera spins in the plane it is.

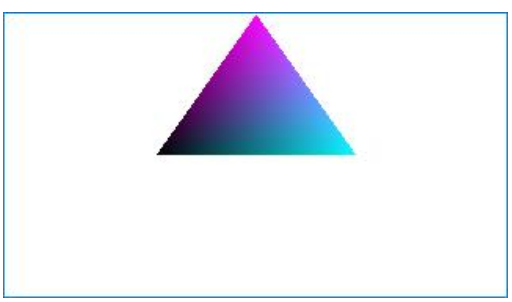
Taking into account of these parameters, suppose the following example:

Example 1

1. The camera is at 1Z, that means is 1 unit in our direction, Position = 0,0,1
2. Points to the coordinates origin., Target = 0,0,0.
3. It does not spin, Up = 0,1,0

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

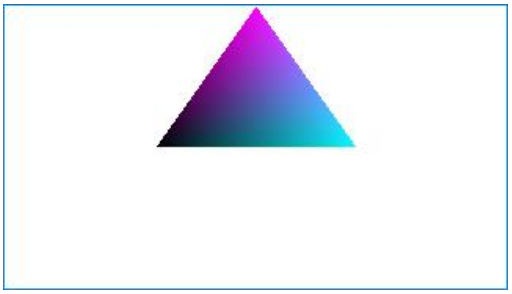
Taking into account this parameters (without considering the projection, neither how the triangle is defined) we have:



Example 2

Camera at 10Z:

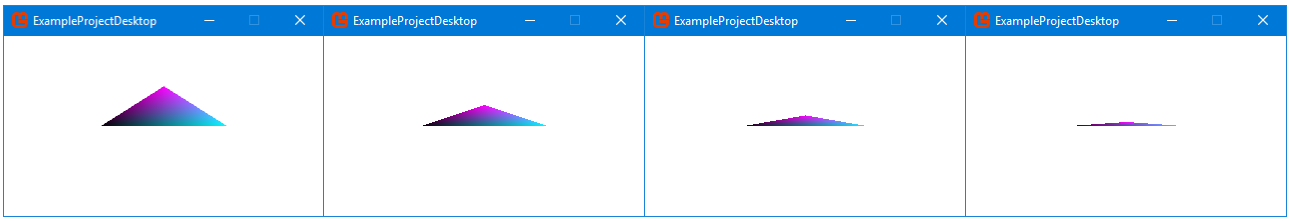
```
camera = new Camera(new Vector3(0, 0, 10), Vector3.Zero, Vector3.Up, -5, 5, 5, -5);
```



Both examples are equal, because we are not using perspective, the purpose of 2.5D is we want to have the same result when changing the position of the camera.

Example 3

We change the camera position Y:




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

The values for offset are 2,4,6,10.

The camera rotation is like ‘inverse exponential reach, it means at the beginning the rotation is fast but then it is really slow until arrives to the horizontal.

¿Why up to 10?

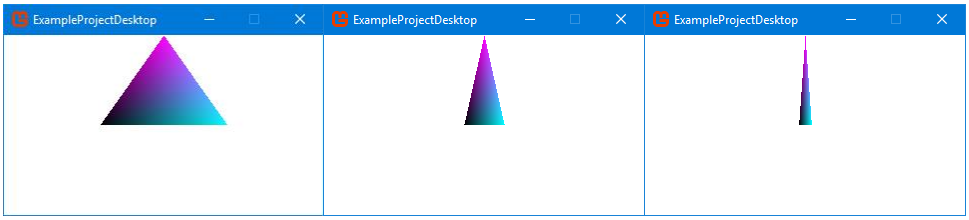
In the projection we are using, we use a far panel of 10, I understand it like the Y will be the Z when arrives to that value.

If we use 50 it will happen in that value

NOTE: If we use smaller values in the projection for this when changing the Y, the shapes are not always well rendered.

Example 4

In this case we change the X position of the camera:



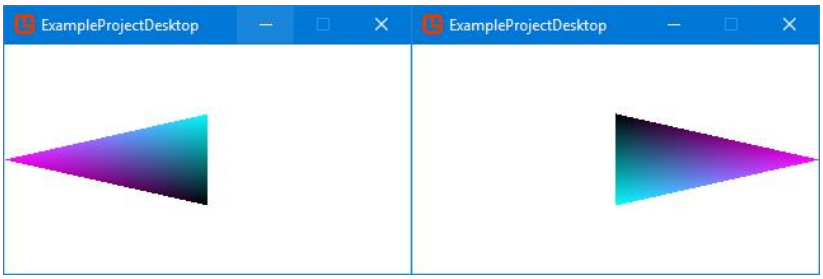
The camera move in the X axis and always points to 0,0,0.

NOTE: If we use a position in the camera with Z=0 the triangle is not rendered.

NOTE: We change the value of the camera target. In this the triangle spins and also translates a bit because we are at 1Z of distance. Show screenshots is not relevant.

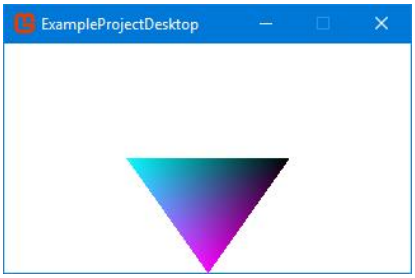
Example 6

Let's change the last vector, the CameraUpVector in the X axis, without adding the Vector Up. Depending on the X value greater or lower than zero, it reflects the triangle.



Example 7

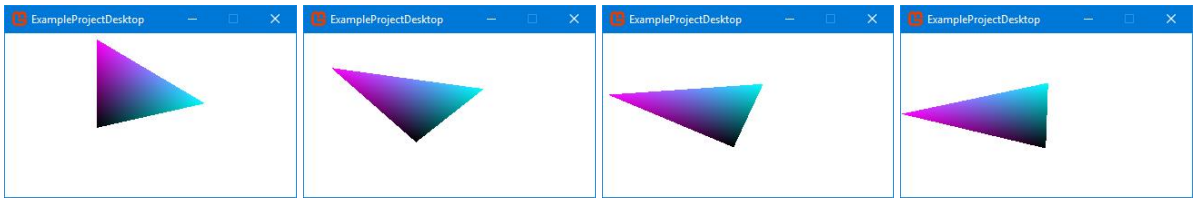
Let's change the CameraUpVector in the Y without adding the Vector Up. In this case is reflected in the Y axis.



Example 8

Finally let's change CameraUpVector in the X axis and adding the Vector Up.

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



We see how it deforms and rotates.

NOTE: Changing the Z axis in the CameraUpVector does not make any result.

Conclusion

At this moment the best parameters are the following:

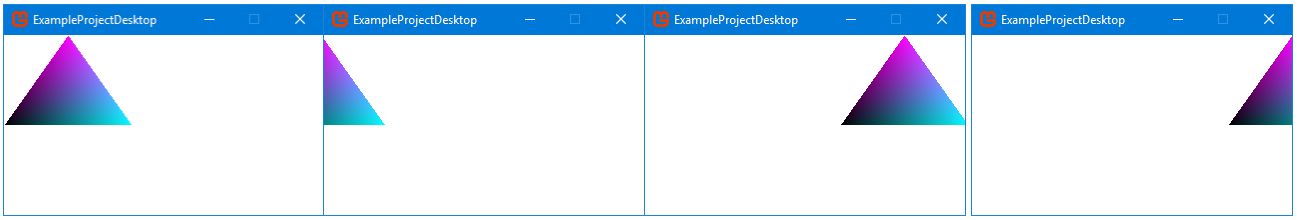
```
camera = new Camera(new Vector3(0, 0, 1), Vector3.Zero, Vector3.Up, ...);
```

Translation in the X axis

Let's manage the movement of the camera, in theory we have to change both values in position and in target vectors.

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

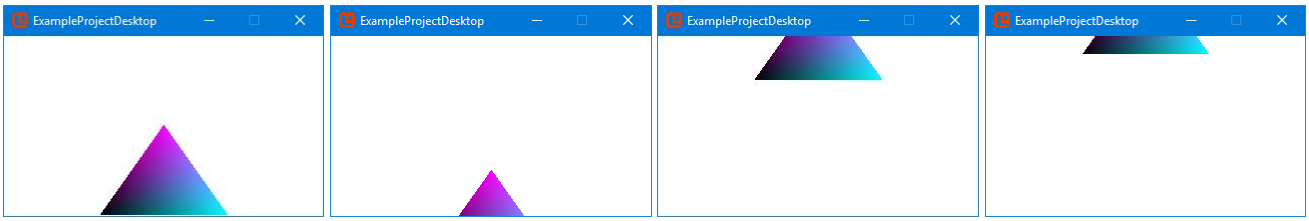
For values of offset in X: 3,5,-3,-5



Translation in the Y axis

For values of offset in the Y axis: 5,7, -2.5, 4

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



NOTE: Changing in the Z axis does not make any difference because the projection we are using.

Conclusion

In a 2.5D environment, we can translate the camera in the X and Y axis but not in the Z axis.

We will see later how we can make the elements look nearer and farer in the Z axis.