

CGameState.cs

JewelThiefLevelDesigner/JewelThief/CGameState.cs

```

namespace JewelThief
{
    /// <summary>
    /// This class is used to store the position and
    /// velocity of a dynamic entity. Note that
    /// both the position and velocity are in units of
    /// pixels.
    /// </summary>
    class CSprite
    {
        public CPoint2f Position = new CPoint2f();
        public CPoint2f Velocity = new CPoint2f();
    }
    class CEnemy : CSprite
    {
        public CPoint2f TargetPosition = new CPoint2f();
    }
}

```

CGameState.cs

JewelThiefLevelDesigner/JewelThief/CGameState.cs

```

/// This class represents a state of the game. It has two main
/// the Player which represents the position and velocity of th
/// user and an array of Enemy states. There is one entry in th
/// for each enemy in the game. It is important to note that th
/// number of enemies must be known when the class is construct
class CGameState
{
    public CSprite Player = new CSprite();
    public CEnemy[] Enemies;
    public CGameState(int num_enemies)
    {
        if(num_enemies > 0) Enemies = new CEnemy[num_enemies];

        for(int i=0; i<num_enemies; i++)
        {
            Enemies[i] = new CEnemy();
        }
    }
}

```

CGameState.cs

```

namespace JewelThief
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow: Window
    {
        public CLevel currentLevel {get; set; } = new CL
        CGameTextures gameTextures;

        public MainWindow()
        {
            InitializeComponent();
            //int width = int.Parse(txtWidth.Text);
            int height = int.Parse(txtHeight.Text);
            gameTextures = new CGameTextures();
            //currentLevel.Resize(width, height);
        }
    }
}

```

CGameState.cs

```

// This function is used as an event handler for
// event. This is used to load a new level. In ad
// the data required for the level it also ensure
// displayed.
private void btnLoad_Click(object sender, RoutedEventArgs)
{
    // Get the directory where the level data is
    // load the data in.
    string fileDir = txtLevelDir.Text;
    currentLevel = CLevelParser.ImportLevel(fileDir)
    gameTextures = CLevelParser.ImportTextures(fi
    txtWidth.Text = currentLevel.Width.ToString()
    txtHeight.Text = currentLevel.Height.ToString()
    RefreshTexturelist();
    RedrawLevel();
}

```

CGameState.cs

```
private void RefreshTexturelist ()
{
    imgWall.Source      = gameTextures.WallTexture;
    imgFloor.Source     = gameTextures.FloorTexture;
    imgEnemy.Source     = gameTextures.EnemyIcon;
    imgPlayer.Source    = gameTextures.PlayerIcon;
    imgGoal.Source      = gameTextures.GoalIcon;
}
```

CGameState.cs

```
private void RedrawLevel ()
{
    if (!gameTextures.IsSet ())
    {
        return;
    }
    // clear any existing children.
    cvsMainScreen.Children.Clear ();
    // Draw the set of wall and floor tiles for this level.
    // level and the goal icon. This is part of the level.
    // we do not expect to change.
    DrawLevel ();
}
```

CGameState.cs

```
// Set up the player to have the correct .bmp
// its initial starting point. The player's position
// as a tile index on the Clevel class, this position
// to a pixel position on the game state.
Image playerIcon = new Image ();
playerIcon.Width = CGameTextures.TILE_SIZE;
playerIcon.Height = CGameTextures.TILE_SIZE;
playerIcon.Source = gameTextures.PlayerIcon;
cvsMainScreen.Children.Add (playerIcon);
Canvas.SetLeft (playerIcon, currentLevel.StartX * TILE_SIZE);
Canvas.SetTop (playerIcon, currentLevel.StartY * TILE_SIZE);
```

CGameState.cs

```
// Create instances of the enemies for display
for (int i=0; i<currentLevel.EnemyPositions.Count; i++)
{
    Image enemyIcon = new Image ();
    enemyIcon.Width = CGameTextures.TILE_SIZE;
    enemyIcon.Height = CGameTextures.TILE_SIZE;
    enemyIcon.Source = gameTextures.EnemyIcon;
    cvsMainScreen.Children.Add (enemyIcon);
    Canvas.SetLeft (enemyIcon, currentLevel.EnemyPositions[i].X * TILE_SIZE);
    Canvas.SetTop (enemyIcon, currentLevel.EnemyPositions[i].Y * TILE_SIZE);
}
}
```

Creating a tool to convert Jewel Thief levels into doom3 levels

- one interesting aspect of open format games is the ability to either share assets or easily convert assets from one game to another
- recall from last week the level1.txt was:

Creating a tool to convert Jewel Thief levels into doom3 levels

```

Height=12
Width=12
FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFWWWWWF
WWWFWFWFWFW
FFFFFFFWFWFW
FFFFFFFWFWFW
FFFFFFFWFWFW
FFFFFFFWFWFW
FWWWWWFWWWF
FFFFFFFWWWFFF
FFFFFFFWWWFFF
row 0
row 11

```

Python tool to convert this format into a doom3 map

```

#!/usr/bin/env python

import sys, getopt

def usage (code):
    print "autoftp [-v] [-p] [-h]"
    sys.exit (code)

filename = ""
verbose = False

```

Python tool to convert this format into a doom3 map

```

def handleOptions ():
    global filename
    try:
        optlist, l = getopt.getopt (sys.argv[1:],
                                    ":vhf:")
    except getopt.GetoptError:
        usage (1)

    for opt in optlist:
        if opt[0] == "-h":
            usage (0)
        if opt[0] == "-v":
            verbose = True
        if opt[0] == "-f":
            filename = opt[1]

```

Python tool to convert this format into a doom3 map

```
def main ():
    handleOptions ()

main ()
```

Python tool to convert this format into a doom3 map

```
#!/usr/bin/env python

import sys, getopt

def usage (code):
    print "w2p [-v][-h][-o outputfile] inputfile"
    sys.exit (code)

inputFile = ""
outputFile = ""
verbose = False
```

Python tool to convert this format into a doom3 map

```
def handleOptions ():
    global inputFile, outputFile, verbose
    try:
        optlist, l = getopt.getopt(sys.argv[1:],
                                   ':vhf:o:')
    except getopt.GetoptError:
        usage(1)

    for opt in optlist:
        if opt[0] == '-h':
            usage (0)
        if opt[0] == '-v':
            verbose = True
        if opt[0] == '-o':
            outputFile = opt[1]
    inputFile = l[0]
```

Python tool to convert this format into a doom3 map

```
def main ():
    handleOptions ()
    print "input =", inputFile, "output =", outputFile

main ()
```

Python tool to convert this format into a doom3 map

```
$ python w2d.py -o foo bar  
$ input = bar output = foo
```