Game Development

Practical 9

Year 4 Semester 2, 2014

Game Story:

Once upon a time there was a good king in a peaceful kingdom. One day a witch came to this kingdom and imprisoned the princess (daughter of the king) in a cave. Then the witch ensured the security of the cave by placing a dragon in front of the cave door.

At that time, there was a super hero called **Agent Windy.** The King made a request from Agent Wendy to survive the princess from the witch. Agent Wendy accepted the challenge and went to save the princess.

Agent Wendy travels through a road that full of obstacles. It is required to go to the cave within 3 minutes. The dragon sends fire balls towards the Agent Wendy. Agent Wendy needs to shoot the fireballs to destroy them. Initially Agent has 5 lives. If fireballs touch the Agent, the lives will be reduced by one. Fairies drop healthy mushrooms from the sky. If the Agent Wendy catches 10 of them, his life will be refilled by one. Same time the witch sends poisonous mushrooms. If the Agent Wendy catches a poisonous mushroom his journey will be delayed by 10 seconds. Finally Agent needs to reach to the dragon and shoot 15 bullets to kill the dragon. Then the Agent Wendy can win the challenge and save the princess.

Step 1: Create a New Project

1. From the **Start** menu, click **All Programs**, click the **XNA Game Studio 4.0** folder, and then click your supported version of Microsoft Visual Studio tools.

2. When the Start Page appears, click the **File** menu, and then click **New Project**.
   A dialog box appears with a tree list on the left pane, marked Project Types.

3. Select the **XNA Game Studio XNA Game Studio 4.0** tree node underneath the **Visual C#** node. A set of available projects appears in the right pane.

4. In the right pane of the dialog box that appears, click **Windows Game (4.0)**, and then type a title for your project (such as "AgentWindy") in the **Name** box.

5. Type a path where you'd like to save your project in the **Location** box, and then click **OK**. After creating a new project, you'll be presented with the code view of your game.

Step 2: View the Code

Some of the hard work has already been done for you. If you build and run your game now, the **GraphicsDeviceManager** will set up your screen size and render a blank screen. Your game will run and update all by itself. It's up to you to insert your own code to make the game more interesting. Much of the code to start and run your game has already been written for you. You can insert your own code now.

- The **Initialize** method is where you can initialize any assets that do not require a **GraphicsDevice** to be initialized.

- The **LoadContent** method is where you load any necessary game assets such as models and textures.
- The **UnloadContent** method is where any game assets can be released. Generally, no extra code is required here, as assets will be released automatically when they are no longer needed.

- The **Update** loop is the best place to update your game logic: move objects around, take player input, and decide the outcome of collisions between objects, and so on.

- The **Draw** loop is the best place to render all of your objects and backgrounds on the screen.

Step 3: Add resources

The next step is to add a graphic that can be drawn on the screen.

Once you have a graphic picked out on your computer, follow these steps.

1. Make sure you can see the Solution Explorer for your project on the right side of the window.

   If you cannot see it, click the **View** menu, and then click **Solution Explorer**.

   When it appears, you will see files associated with your project in a tree structure. Inside the tree, you will see a node named **Content**.

2. Right-click the **Content** node, click **Add**, click **Existing Item**, and then browse to your graphic.

   If you can't see any files, make sure you change the **Files of type** selection box to read **Texture Files**.

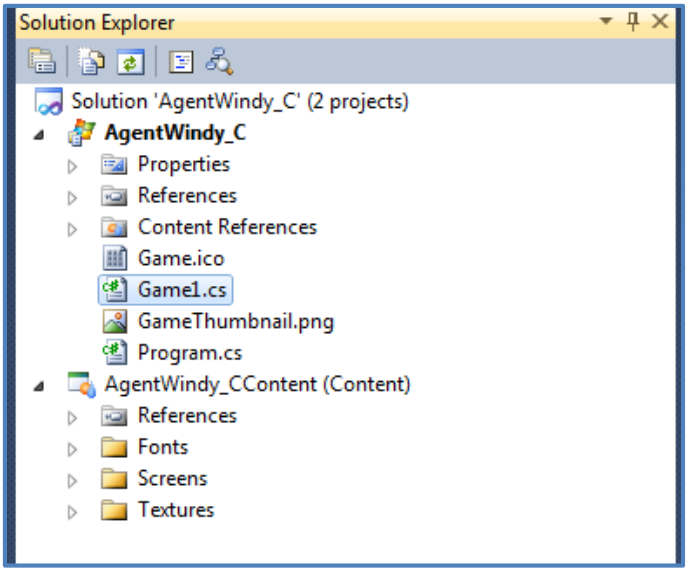
3. Click the graphic file, and then click **Add**.

An entry for the graphic file will appear in Solution Explorer.

4. Click the entry for the graphic in the Solution Explorer. If you do not see the entry, ensure the **Content** node is expanded by clicking the small plus sign (+) to the left of the node, and then click on the entry that appears underneath the **Content** node.

When you add a graphic file, it is added automatically to the XNA Framework Content Pipeline. This allows you to quickly and easily load the graphic into your game.

5. Create 3 folders inside the Content node to store the necessary resources of the game.



6. Now store necessary game assets inside the relevant folders.

Ex: Background, start screen, won screen, game over screen inside the Screen folder.

Fonts inside the font folder.

Dragon , agent windy , mushrooms, fireballs, bullets etc inside the texture folder.

Step 4 : Load the game assets

Now click on the Game1.cs class and load the above game assets to the game.

In XNA we declare 2D images as Texture 2D objects, 3D images as Texture 3D objects and fonts as sprite fonts. Accordingly declare the game sprites and fonts as follows.

Inside the Game1.cs class soon after the following lines declare the texture 2D and sprite font objects.

```
GraphicsDeviceManager graphics;
SpriteBatch spriteBatch;
```

```
amespace AgentWindy_C

  /// <summary>
  /// This is the main type for your game
  /// </summary>
  public class Game1 : Microsoft.Xna.Framework.Game
  {
      GraphicsDeviceManager graphics;
      SpriteBatch spriteBatch;

      Texture2D agentTex, DragonTex, helthMushroomTex, poisonMushroomTex, fireballTex, bulletTex, backgroundTex;
      SpriteFont life, timeLeft, counDown;

      Texture2D startScreen, gameOverScreen, youWonScreen;

      public Game1()
      {
          graphics = new GraphicsDeviceManager(this);
          Content.RootDirectory = "Content";
      }
```

Then go inside the LoadContent() method and include the following lines to load the game assets.

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);


    life = Content.Load<SpriteFont>("Fonts\\ControlFont");
    counDown = Content.Load<SpriteFont>("Fonts\\BigControlFont");

    agentTex = Content.Load<Texture2D>("Textures\\windy");
    DragonTex = Content.Load<Texture2D>("Textures\\dragon");
    helthMushroomTex = Content.Load<Texture2D>("Textures\\healthy_mushroom");
    poisonMushroomTex = Content.Load<Texture2D>("Textures\\poisonus_mushroom");
    fireballTex = Content.Load<Texture2D>("Textures\\fireball");
    bulletTex = Content.Load<Texture2D>("Textures\\bullet");
    backgroundTex = Content.Load<Texture2D>("Screens\\background");



    startScreen = Content.Load<Texture2D>("Screens\\start");
    gameOverScreen = Content.Load<Texture2D>("Screens\\go");
    youWonScreen = Content.Load<Texture2D>("Screens\\won");
```

Then declare the positions of the objects inside Game1.cs after the line.

```
Texture2D startScreen, gameOverScreen, youWonScreen;
```

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;

    Texture2D agentTex, DragonTex, helthMushroomTex, poisonMushroomTex, fireballTex, bulletTex, backgroundTex;
    SpriteFont life, timeLeft, counDown;

    Texture2D startScreen, gameOverScreen, youWonScreen;

    Rectangle agent, Dragon, bullet;
    Rectangle helthMushroom1, helthMushroom2;
    Rectangle poisonMushroom1, poisonMushroom2, poisonMushroom3;
    Rectangle fireball;

    Vector2 background;
    Vector2 timeRemain, countDownPosition;
```

Then declare the positions and sizes inside the LoadContent() method.

```
agent = new Rectangle(50, 320, 50, 75);
Dragon = new Rectangle(850, 290, 80, 105);
fireball = new Rectangle(850, 330, 30, 30);
bullet = new Rectangle(50, 330, 30, 30);

helthMushroom1 = new Rectangle(500, 10, 40, 40);
helthMushroom2 = new Rectangle(700, 20, 40, 40);

poisonMushroom1 = new Rectangle(300, 30, 40, 40);
poisonMushroom3 = new Rectangle(400, 30, 40, 40);
poisonMushroom3 = new Rectangle(600, 30, 40, 40);

background = new Vector2(0.0f, 0.0f);
timeRemain = new Vector2(800.0f, 50.0f);
countDownPosition = new Vector2(50.0f, 50.0f);
```

Declare and initialize the necessary variables inside the Game1.cs

```
int fireTime,startTime,secondPassed,timeRemains;
int agentLife = 5;
int dragonLife = 15;

int lifeBucket = 0;

int playerRunningSpeed = 2;
```

Declare the starting game screen after the above mentioned lines.

```
string gameState = "Start";
```

Inside the constructor define the screen width and height according to your background screen.

```
public Game1()
{
    graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";
    graphics.PreferredBackBufferHeight = 580;
    graphics.PreferredBackBufferWidth = 1024;
}
```

We need to include a countdown timer to the screen. Therefore declare the following statements inside LoadContent() method.

```
startTime = 180 * 1000;
secondPassed = 0;
timeRemains = 0;
```

Step 5: Enable Keyboard Inputs

We need to control the game by obtaining user inputs from the keyboard.

Therefore inside the Game1.cs class declare an object from the keyboard state as follows.

```
KeyboardState keyState;
```

The game logic should be included inside the Update() method.

```
    protected override void Update(GameTime gameTime)
    {
        keyState = Keyboard.GetState();
```

We will start the game after pressing space key. Include the following code inside Update() method.

```
if (gameState == "Start" && keyState.IsKeyDown(Keys.Space))
{
                gameState = "Running";
}
```

Define the other keys to enable the movement of the agent.

Left arrow key - Left movement
Right arrow key – Right movement
Up arrow key – Up
Down arrow key – Down

Include the following code inside the Update() method.

```
if (keyState.IsKeyDown(Keys.Left))
            {
                agent.X -= playerRunningSpeed;
                bullet.X = agent.X;
```

```
            }

            if (keyState.IsKeyDown(Keys.Right))
            {
                agent.X += playerRunningSpeed;
                bullet.X = agent.X;
            }

            if (keyState.IsKeyDown(Keys.Up))
            {
                agent.Y -= playerRunningSpeed;
                bullet.Y = agent.Y;
            }

            if (keyState.IsKeyDown(Keys.Down))
            {
                agent.Y += playerRunningSpeed;
                bullet.Y = agent.Y;
            }
```

Enable the feature of dragon firing towards the agent.

Include the following code inside the Update() method.

```
fireball.X -= 2;

if (fireTime % 500 == 0)
{
          fireball = new Rectangle(850, 380, 30, 30);
}
```

Now we will include two types of mushrooms.
Include the following code inside the Update() method.

```
helthMushroom1.Y += 2;
helthMushroom2.Y += 3;
poisonMushroom1.Y += 4;
poisonMushroom2.Y += 3;
poisonMushroom3.Y += 2;



if (fireTime % 2200 == 0)
 {
            helthMushroom1 = new Rectangle(500, 10, 60, 60);
            helthMushroom2 = new Rectangle(700, 20, 60, 60);
```

```
            poisonMushroom1 = new Rectangle(300, 30, 60, 60);
            poisonMushroom2 = new Rectangle(400, 30, 60, 60);
            poisonMushroom3 = new Rectangle(600, 30, 60, 60);
}


if (fireTime % 3200 == 0)
{
            helthMushroom1 = new Rectangle(500, 10, 60, 60);
            helthMushroom2 = new Rectangle(700, 20, 60, 60);

}
```

Include the firing time after the above code segment.

```
fireTime += gameTime.ElapsedGameTime.Milliseconds;
secondPassed += gameTime.ElapsedGameTime.Milliseconds;
timeRemains = startTime - fireTime;
```

If the remaining time finish we need to exit from the screen.

```
if (timeRemains < 0)
{
          this.Exit();
}
```

Enable the shooting facility of the agent with the key S.

```
if (keyState.IsKeyDown(Keys.S))
{
                bullet.X += 10;
}
```

Handle the collisions of the game fire balls with the agent. If the bullet intersects the dragon dragon's life will be lose by one.

```
if (fireball.Intersects(bullet))
{
                bullet.X = agent.X;
                fireball = new Rectangle(850, 380, 30, 30);
 }
```

```
 if (Dragon.Intersects(bullet))
 {
                dragonLife--;
                bullet = new Rectangle(850, 380, 30, 30);

 }
```

Increase the number of lives of the agent when the agent grabs healthy mushrooms.

```
if (agent.Intersects(helthMushroom1) ||
agent.Intersects(helthMushroom2))
{
                lifeBucket++;
                helthMushroom1 = new Rectangle(500, 10, 40, 40);
                helthMushroom2 = new Rectangle(700, 20, 40, 40);
}

if (lifeBucket == 10)
{
                agentLife++;
                lifeBucket = 0;
}
```

Decrease the number of lives of the agent when the agent grabs poisonous mushrooms or hits fireballs.

```
  if (agent.Intersects(fireball))
  {
                agentLife--;
  }


 if (agent.Intersects(poisonMushroom1) ||
agent.Intersects(poisonMushroom2) ||
agent.Intersects(poisonMushroom3))
{
                agentLife--;
                playerRunningSpeed = 0;
                poisonMushroom1 = new Rectangle(300, 30, 40, 40);
                poisonMushroom2 = new Rectangle(400, 30, 40, 40);
                poisonMushroom3 = new Rectangle(600, 30, 40, 40);

}
```

Now we'll move to won state of the game or game over state of the game.

If the agent is killed by the dragon, the player will move to the game over state. If the dragon is killed by the agent, player will move to the won state.

```
if (agentLife <= 0)
{
                gameState = "Over";
}
if (dragonLife <= 0)
{
                gameState = "Won";
}
```

Finally we will design different game screens in different situations. Include the following code segments inside Draw() method.

First of all we'll include the start screen. You can include any image as your start screen.

```
if (gameState == "Start")
{
                spriteBatch.Begin();
                spriteBatch.Draw(startScreen, new Vector2(0, 0),
Color.White);
                spriteBatch.End();
}
```

Then we will move to the running state. When the game is running we need to show all the game objects inside the screen.

```
if (gameState == "Running")
{
                spriteBatch.Begin();
                spriteBatch.Draw(backgroundTex, background, Color.White);

                spriteBatch.DrawString(life, "Agent Lifes : " +
agentLife.ToString(), timeRemain, Color.Black);
                spriteBatch.DrawString(life, "Dragon Lifes :" +
dragonLife.ToString(), new Vector2(800.0f, 100.0f), Color.Black);
                spriteBatch.DrawString(counDown, timeRemains.ToString(),
countDownPosition, Color.Red);
                spriteBatch.DrawString(life, "Millyseconds is to defeat the
Dragon..!!!", new Vector2(50.0f, 120.0f), Color.Black);



                spriteBatch.Draw(bulletTex, bullet, Color.White);
                spriteBatch.Draw(agentTex, agent, Color.White);
                spriteBatch.Draw(DragonTex, Dragon, Color.White);
                spriteBatch.Draw(fireballTex, fireball, Color.White);

                spriteBatch.Draw(helthMushroomTex, helthMushroom1,
Color.White);
                spriteBatch.Draw(helthMushroomTex, helthMushroom2,
Color.White);

                spriteBatch.Draw(poisonMushroomTex, poisonMushroom1,
Color.White);
                spriteBatch.Draw(poisonMushroomTex, poisonMushroom2,
Color.White);
                spriteBatch.Draw(poisonMushroomTex, poisonMushroom3,
Color.White);


                spriteBatch.End();
}
```

Now it's time to go for WON state.

```
if (gameState == "Won")
{
                spriteBatch.Begin();
                spriteBatch.Draw(youWonScreen, new Vector2(0, 0),
Color.White);
                spriteBatch.End();
}
```

Finally we will include the game over state.

```
if (gameState == "Over")
{
                spriteBatch.Begin();
                spriteBatch.Draw(gameOverScreen, new Vector2(0, 0),
Color.White);
                spriteBatch.End();
}
```

Now run the game and play. ☺