HONG KONG INSTITUTE OF VOCATIONAL EDUCATION

Laboratory 6: Simple 2D Game – Phase V (User Input and Audio)

Module Intended Learning Outcome (#2, #5):
On completion of the module, students are expected to be able to:
- develop 2D and 3D graphics programs for general gaming purposes;
- identify and apply appropriate user input methods for graphics and/or game programs

Lesson Intended Learning Outcome:
On completion of this lab, students are expected to be able to:
- Create a simple game with user interaction for Windows using MonoGame and Visual Studio.
- Compile, debug and execute a game program using MonoGame and Visual Studio.

In this lab, you will use your mouse instead of the keyboard to control the game. The game play will also be different from that of Lab 5. You will apply your mouse click to destroy the falling rocks so to avoid them from hitting your running man.

TASK 1 – Catch the Rocks

1. Start modifying the project from Lab 4 project. Your running man should run from left to right automatically at the very beginning.

2. Add a **ButtonState** variable to store the state of left mouse button in the last update. We assume the button is initially released.
```
ButtonState LastMouseButtonState = ButtonState.Released;
```

3. Add a **Vector2** variable ***mouse*** to store the mouse cursor position and a **bool** variable ***clicked*** to store if the mouse has been clicked.
4. As in Lab5, add two more variables to store the score (***int score***) and remaining life (***int life***) in the module level. Initialize them to 0 and 5 respectively.

5. Also as in Lab 5 (Task 2), add a new **Sprite Font** in your Content Pipeline Tool. Add a **SpriteFont** variable to store the sprite font and load the content of the font to the variable in the given ***LoadContent()*** method.
6. In order to show you mouse cursor, you need to draw a cursor image at your mouse position or using the default Windows mouse cursor. To use the default one, type in the following statement in ***Initialize()*** method:
```
IsMouseVisible = true;
```
We will choose to use our own image in this lab. Add a **Texture2D** variable to store the cursor image. Insert the given cursor image to your **Content** folder in your Content Pipeline Tool and load the content to the variable in the given ***LoadContent()*** method. The image will be drawn later.
7. Add a private method ***UpdateInput()*** method into your *Game1.cs* file to handle the mouse input. The function should have the following header:
```
private void UpdateInput()
```
8. In your ***UpdateInput()*** method, you should do the followings:
   i) Get and store the ***MouseState*** information from your ***Mouse*** object.
   ```
   MouseState ms = Mouse.GetState();
   ```
   ii) Store the mouse position from the **MouseState** to your **Vector2** variable.
   iii) Check if your mouse button is pressed in this frame, but not in the last frame.

```
            if (ms.LeftButton == ButtonState.Pressed &&
                  ms.LeftButton != LastMouseButtonState) { /* DO SOMETHING
*/ }
```

iv) If the mouse button has been clicked, set the variable ***clicked*** to **true**; otherwise, set it to **false**.

v) Store the current state of left button to the **ButtonState** variable ***LastMouseButtonState***.

9. Call ***UpdateInput()*** method in your ***Update()*** method.
10. Inside your ***CheckCollison()*** method, for each of your rocks, check if the mouse click on the rock by checking if the pixel clicked has a color with alpha value != 0. The process is similar to that in ***PixelCollison()*** method.

```
      // Transform the mouse position to rock coordinates using invert
      Vector2 mouseInRock = Vector2.Transform(mouse,
Matrix.Invert(rockTransform));
      int xB = (int)Math.Round(mouseInRock.X);
      int yB = (int)Math.Round(mouseInRock.Y);
      // if the pixel is within the rock's sprite rectangle
      if (0 <= xB && xB < r.texture.Width && 0 <= yB && yB < r.texture.Height)
{
            // Get the color of that pixel and check if alpha channel = 0
            Color color = r.ColorData[xB + yB * r.texture.Width];
            if (color.A != 0) {
                  // mouse hit rock, reset the rock position, add mark
            }
      }
```

11. Add 1 to score when the mouse hit a stone and reduce life by 1 when a rock hit the man. Remember to include the collision delay (Lab 5 Task 3) in this game.
12. Draw the cursor image onto the screen at (***mouse.x***, ***mouse.y***) using ***Draw()*** method in your ***SpriteBatch.***

```
  spriteBatch.Draw( cursorImage, mouse, Color.White );
```

13. Draw texts showing the score and remaining life onto the screen using ***DrawString()*** method in your ***SpriteBatch*** object with the specified sprite font as in the last lab.

```
  Rectangle rect = GraphicsDevice.Viewport.Bounds;
  spriteBatch.DrawString( font, "Score: " + score,
              new Vector2(20, rect.Height - 30), Color.White);
  string message = "Life Remain: " + life;
  spriteBatch.DrawString(font, message,
        new Vector2(rect.Width - font.MeasureString(message).X - 20,
        rect.Height - 30), Color.White);
```

14. Stop updating your game if the remaining life is 0. Show a "Game Over" message near the center of the screen.

TASK 2 – Adding Sound

To make your game more interesting, you may like to add some sound effect to your game project. It is easy to do.

1. Similar to adding images, add the given sound files to the project through the Content pipeline tool under a new subfolder "Sounds".
2. Add **SoundEffect** variables for storing the sounds (wav) respectively. To use **SoundEffect**

class, you will also need to have the following using statement:

```
using Microsoft.Xna.Framework.Media;
```

3. In your ***LoadContent()*** method, load the sound files using ***Content.Load()*** method.

4. After loading the files into your project, you can play the sound anywhere in your code by simply calling **SoundEffect**’s ***Play()*** method.
5. Try use the gun shot sound for every click on the mouse button, explosion sound for every rock destroyed, and scream sound for every rock that hits the man.
6. In addition to the above sound effects, I have also included an mp3 file in the given zip file.
   - Add a **Song** variable to store the mp3 background music. However, to use the **Song** class, you need to have the following using statement first:

     ```
     using Microsoft.Xna.Framework.Media;
     ```
   - In your ***LoadContent()*** method, load the mp3 files using ***Content.Load()*** method.
   - Right after you have loaded the file, you can play it in your game using the following statements. It will keep repeatedly playing till the game ends.

     ```
     MediaPlayer.Play(bgMusic);
     MediaPlayer.IsRepeating = true;
     ```

   Notes: You can always find some loyalty free music background suitable for your game on the web. My song is from http://soundimage.org. The site requires you to attribute the author when you use his music/sound effects. So, please add a line of credit (shown in the first 5 seconds) in your gameplay as follows:

```
            if(gameTime.TotalGameTime.TotalSeconds<5) // first 5 seconds only
                spriteBatch.DrawString(font, "Music by Eric Matyas
(soundimage.org)",
                new Vector2(20, 20), Color.White, 0, Vector2.Zero,
                0.75f, SpriteEffects.None, 0); // 0.75f for smaller font
```