Lab 11 – XNA Input

Instructions: Complete each problem. If you're struggling with a problem, feel free to ask questions on the class forum.

This lab is optional, but it gives you valuable programming experience. You should definitely complete the lab if you can.

Getting Started

Double-click the index file in the Help folder and click the ExplodingTeddies link in the pane on the left; this is the documentation for the classes I provided to you in the ExplodingTeddies dll.

Problem 1 – Create a project, add content, add moving teddy bear

Start up the IDE and create a new MonoGame Windows Project (or appropriate MonoGame project for your OS) named Lab11. Save the project in a reasonable location on the computer.

Build content for the teddy bear image and the explosion image from the zip file and add the content to your project.

Copy the ExplodingTeddies.dll file you extracted from the zip file you downloaded into the Lab11 project folder (the folder that contains the Game1.cs file). Add a reference to the dll using the same steps you used in Lab 10.

Add the following code below the `using` statements already in the class:

```
using ExplodingTeddies;
```

Add two constants to the `Game1` class just above the declaration of the `graphics` variable:

```
public const int WindowWidth = 800;
public const int WindowHeight = 600;
```

Just below the line that says `SpriteBatch spriteBatch;` near the top of the `Game1` class, add variable declarations for a `TeddyBear` object and an `Explosion` object.

Just below your new variables, declare a variable for a `Random` object and call the `Random` constructor to create an object for the variable. You'll need to add a using statement for the `System` namespace to get this to compile.

Just below the line that says `Content.RootDirectory = "Content";` near the top of the `Game1` constructor, add the following two lines of code:

```
graphics.PreferredBackBufferWidth = WindowWidth;
graphics.PreferredBackBufferHeight = WindowHeight;
```

Just below those two lines, add the following line to make the mouse pointer visible:

```
IsMouseVisible = true;
```

In the `Game1 LoadContent` method, replace the comment that says

```
// TODO: use this.Content to load your game content here
```

with a comment and the code to create a new teddy bear object using the `TeddyBear` constructor that gives the teddy bears a specific velocity. You should generate random x and y components of the velocity (as floats) using the random variable you declared above. You should use the `Random NextDouble` method, typecasting to float and multiplying by some reasonable scale factor to give the teddy bear a reasonable speed.

After you have those random velocity components, you can create a new `Vector2` object to pass in as the last argument to the `TeddyBear` constructor.

Add code to create an Explosion object using the `Explosion` constructor.

Add code to the `Game1 Draw` method to have the teddy bear and explosion draw themselves.

Add code to the `Game1 Update` method to have the teddy bear and explosion update themselves.

Problem 2 – Explode the teddy bear when clicked

Add code to the `Game1 Update` method to check if the mouse is over the teddy bear and the left mouse button is pressed. If both those conditions are true, make the teddy bear inactive and start playing the explosion at the center of the teddy bear.

Problem 3 – Spawning a new teddy bear

This portion of the lab uses an Xbox 360 controller. If you don't have a 360 controller, do this portion using the right and middle buttons on the mouse instead.

Add code to replace the current teddy bear with a newly-spawned teddy bear every time the A button is pressed. The behavior you'll see is that the teddy bear will appear to jump when the A button is pressed; this happens because the teddy bear variable now refers to the new teddy bear object and the old teddy bear object is now garbage to be collected.

Comment out the code from Problem 2. Add code to check if the B button is pressed. If it is, start playing the explosion at the center of the teddy bear and replace the current teddy bear with a newly-spawned teddy bear.

To spawn a new teddy bear, your new code needs to:

- Generate a random x for the new teddy bear
- Generate a random y for the new teddy bear
- Generate a random velocity x for the new teddy bear
- Generate a random velocity y for the new teddy bear
- Call the TeddyBear constructor to put a new TeddyBear object into your teddy bear variable

CAUTION: Making the A and B buttons work properly is harder than you may think! You only want to respond to the FIRST time a button is pressed then wait until it's been released before you respond to it again. You'll have to figure out how to make it work that way. Hint: keeping track of the previous state of each button will help you make this all work properly.