Lab 16 – XNA Audio With XACT

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.

Problem 1 – Create an XNA project and build an XACT project

Start up the IDE and create a new Windows Game (4.0) project named Lab16. Save the project in a reasonable location on the computer.

The classes we need to use to play our sounds are contained in the Microsoft.Xna.Framework.Xact dll, which isn't included as a reference in the template Windows Game (4.0) project. To add a reference to that dll, right click References in the code project, then select Add Reference. Select the .NET tab (all the way on the left) in the Add Reference dialog box that pops up, scroll down to the Xna namespaces, select the Microsoft.Xna.Framework.Xact namespace, and click the OK button. Save the project.

Download or create 4 different wav files and save them in the Lab16Content folder.

Start up XACT from the Tools folder that’s located within the XNA installation folder.

Create new wave and sound banks in the project.

Save the XACT project in the Lab16Content folder.

Tile the windows horizontally.

Add the 4 wav files to the project wave bank window.

Copy the files in the wave bank window to the sound name pane of the sound bank window.

Copy the sounds from the sound name pane to the cue name pane of the sound bank window.

Rename the cues to upperLeft, upperRight, lowerLeft, and lowerRight.

Save the project.

Problem 2 – Add audio content and load audio content

Close XACT.

Start up the IDE. Right click the Lab16Content project, and select Add -> Existing Item … Select the .xap project file (mine is called Lab16Audio.xap), and click Add.

Add `AudioEngine`, `WaveBank`, and `SoundBank` fields at the top of the `Game1` class.

In the `Game1` `LoadContent` method, add code to load the audio engine, wave bank, and sound bank content.

Problem 3 – Play cues for left clicks on screen

Make the mouse visible in the `Game1` constructor.

In the `Game1` `Update` method, add code to get the current mouse state.

In the `Game1` `Update` method, add an if statement to check whether the left mouse button is pressed. If it is, play the appropriate cue based on which quadrant (upper left, upper right, lower left, or lower right) the mouse is in.

This may seem to work if you click the mouse normally. Now just hold the left mouse button down – the game starts playing the cue on every update, leading to lots of nasty overlap. Let's fix this to actually use mouse clicks, not mouse button presses.

Add a field to the `Game1` class to tell whether a mouse click was started and initialize the field to `false`.

In the `Game1` `Update` method, add an if statement right after you get the mouse state. If the left mouse button is pressed and the click started flag is `false`, set the click started flag to `true`. Otherwise (else if), if the left mouse button is released and the click started flag is `true`, set the click started flag to `false` and move the code that plays a cue based on the quadrant into this else if clause.