GAMES PROGRAMMING 1

## ALASDAIR GIBSON

## S1111320

# DISCLAIMER

I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award.

SIGNATURE:

Code Explanation

# Main

Main.cpp contains most of the game’s coding, and handles the core gameplay elements as well as organizing the DirectX rendering process.

The program begins by initializing variables for game elements such as lives and points, the speed of the ball, the strings to hold the in-game HUD text in, the sound objects to play audio from, and vectors to control the position and movement of the player’s paddle and ball. Also, references are created to the Direct3D manager, the DirectX manager and the Scene manager.

Next is the LRESULT CALLBACK WndProc, which handles window messages to the program such as key input (through virtual keys like VK\_ESCAPE) and messages to close the game (WM\_CLOSE).

InitWindow comes afterwards, and this Boolean function registers the window class for the application and creates the window for it.

WinMain is where the bulk of the game takes place, and is also the main entry point for windows applications. It initializes the windows, the D3D, DX, Sprite and Scene managers and sets up all the game elements that weren’t defined at the beginning of the Main file (Note that when initializing the Scene manager, this also creates all the necessary Brick objects). This includes loading the custom font (“pix”) into memory, creating the Player paddle and ball sprites, printing the lives/points/level counter strings and setting up the sprite matrices for the ball and paddle.

After checking if the message posted to the window is not ‘quit’, the core rendering and game logic takes place. For the main menu and end screens, this only involves setting up the DirectX renderer and then showing any necessary strings on screen, but for the main gameplay (levels 1-5) it expands to cover input from the left and right keys, limiting the framerate of the rendering, moving the paddle and the ball and updating their respective matrices, and handling collision between the ball, the paddle, and the bricks. This also includes keeping tally of score and lives, and progressing to the next level once all bricks are destroyed or ending the game if the player loses all their lives. Between these are calls to the audio player object (cXAudio) for small sound effects when these events happen. Finally rendering is done for all the brick, paddle and ball sprites, and the points/lives/level text is printed onscreen.

Code Explanation

# Scene Manager

The cSceneManager.cpp file is tasked with identifying the different stages of the game, and keeping track of all the brick objects for each level of the game as well as background images and a count of all the brick objects in the currently selected array. This is done with an enumerated type defined in the cSceneManager.h file, ‘SceneID’, which has 7 types: MENU, LVL1 … LVL5, and END.

When the manager is initialized by calling initializeScnMgr, it sets the current scene to ‘MENU’, sets the current background to the MenuBG.png file and then generates each level’s array of bricks. These are unfortunately incomplete as the game becomes extremely slow when any more than 6 bricks are rendered at a time.

The NextScene method increments the currently loaded scene, and determines which background image matches the scene. If it reaches the END scene, it will loop back to the beginning of the game.

## Bricks

The cBrick class is used to represent a single breakable brick within the game. It takes a position and a set amount of ‘hits’, which means the player has to bounce the ball off the brick that number of times in order to destroy it. cBrick inherits from cSprite, so its variables and functions are very similar. The main difference is the ‘hitbrick’ method, which decrements the number of hits left on the brick as well as changing what sprite is displayed to represent the brick object – Green is one hit, Yellow two, and Red three.

Once the brick has been hit enough times, its ‘Alive’ variable toggles from true to false, and the brick is no longer rendered or checked for collisions in the main game procedure.

References

## Coding assistance

<http://stackoverflow.com/>

<http://www.enel.ucalgary.ca/People/Norman/enel315_winter1997/enum_types/>

<http://www.cplusplus.com/forum/beginner/44859/>

## Sprite asset creation

<http://www.getpaint.net/>

## Audio asset creation

<http://audacity.sourceforge.net/>

<http://lmms.sourceforge.net/>

This list does not include material provided as part of this course.

Storyboards

