Team 9: Matt Schwartz, Michael Passaloukos, Richard Lage
Assignment 2 - Design Document
02/11/2014

Overall Design

We plan on implementing a game where a cannon launches balls in random directions towards the player and he/she has to hit them back to the far wall. Difficulty levels and different game modes will change the speed and/or number of balls present. The player will be a paddle that can move side to side and up and down. The GUI will just have a score (+x points whenever a ball is hit with parts of the wall being worth more points than others and possibly a hit streak that gives bonus points and is reset when the player misses a ball) and difficulty level/game mode so the player knows that information. Sounds will be played when the ball is fired from the cannon, collides with something (wall or paddle) and when the player misses the ball (whiff).

For ball metrics, http://www.racquetresearch.com/coeffici.htm tells us the racquet has a restitution coefficient of 0.85 and http://arxiv.org/abs/1110.3989 suggests that a racquetball has a coefficient of 0.80. We're going to have to play with some forces exerted by the player when he/she hits the ball with it. This could also be something for customization for the player - maybe the player can choose to play as Superman, who exerts more than 2500N of force.

The game would have multiple game modes, if time allows:

- Free Play - just see what you can get your score to. Getting this mode to work will be the main goal for the project.
- Timed Play - how high can you get your streak in X minutes?
- Fast & Furious - balls going really fast; can you last X minutes? You start with Y points and lose Z points when you miss a ball.
- Mission Mode - play several missions such as getting over a certain amount of points in the time limit or hitting specific parts of the wall

Software Architecture and Plan

- Graphics & GUI
    - Resource management, animation, textures, scene management
    - GUI functionality - Score will be displayed in the top-left corner of the screen, along with hit streak, which resets if the player misses the ball
- Sound
    - Finding the resources to use, playing sounds in the game
    - Need at least 3 sound effects: when the ball is fired from the cannon, when the ball collides with an object or wall and when the player misses a ball
- Physics
    - Collision detection between the ball and an object or wall
    - Accurate ball motion (including gravity and forces from the player's paddle)
- Input
    - Player controls (GUI mapping, controls for moving camera, turning off/on sound, other buttons?)

Menus and a "pretty GUI" are the least of our concern and will be completed only if other goals have been met. Texture maps and detailed scene management are slightly higher priority and likewise have less predicted amount of work to implement. Sound (SDL) and physics (bullet) engines both have libraries we can utilize and should be a mild amount of work (after learning how to implement them) to incorporate. We will know how to scale back only after we implement the basic features.

Main game loop:

- Check for user input
- Physics simulation
- Play sounds
- Render scene
    - Animations
    - Render new locations of objects

Object Structures

Cannon

- Position vector
- Texture map/mesh
- Direction vector
- Speed with which it shoots the balls

Ball

- Position vector {x,y,z} with respect to the room
- Size of the ball (i.e., its radius)
- move() method that takes into account force, direction vectors and current position and

its restitution coefficient
- Texture mapping the ball

Player - floating paddle
- Strength - with how much force does the player hit the ball?
- Position vector {x,y,z}
- Paddle mesh/texture

Room
- Dimensions - where the walls are located and their sizes, necessary for detecting collisions
- Where lights are located
- Texture mapping for the walls of the room

We would have a class called ObjectManager (or the like) which would keep track of the state of the objects (player, ball) within a room and each of the objects would keep track of their own internal state, as mentioned above in the structures (e.g., a player object keeps track of its own location, but the ObjectManager would keep track of updating a player's location).

Division of Labor

We will be using github to help keep our work current and easily accessible by all members for quick debugging and testing during development from anywhere.

Matt Schwartz

- Physics simulation
- Learning how to use Bullet with Ogre and creating a simple interface for the other members to use for the objects in the game that require physics (the racquetball for example)

Michael Passaloukos

- Resource management - sound effects, texture maps
- Sound effects
- Player input/controls

Richard Lage

- Scene management (player interactions with other objects in the scene)
- Implementing the GUI
- Rendering