Jason Sekhon - A00952521
Renz Dionglay
Morris Arroyo
Eric Lin

Objective

Two players take turns launching projectiles at each other. Hit the opponent and deplete their health bar to win.

Software Design

Imported Models

These are models that we made in blender and imported using an objloader. Models have data representing vertex positions, colour, texture coordinates, and the normal. Also contains values for ambient, diffuse, specular, and shininess. These values are used by a models corresponding node or pnode gameobject.

Angler.m

Data for use in creating the angler model. Model made in blender and imported with an objloader. Represents the angle and velocity representation in the gamescene

Parachute.m

Data for use in creating the parachute model. Model made in blender and imported with an objloader. Represents the parachute at the start of each round when the tanks are falling down.

Player1Wins.m

Data for use in creating the player 1 wins text. Made in blender and imported with an objloader. Represents the text displayed in the game over scene when player one wins.

Player2Wins.m

Data for use in creating the player 2 wins text. Made in blender and imported with an objloader. Represents the text displayed in the game over scene when player two wins.

Tank1.m

Data for use in creating the tank for player 1. Made in blender and imported with an objloader. Represents the red tank controlled by player one.

Tank2.m

Data for use in creating the tank for player 2. Made in blender and imported with an objloader. Represents the green tank controlled by player two.

Floor.m

Data for use in creating the floor model. Made in blender and imported with an objloader. Represents the floor on which the tanks settle and move on.

Nodes

Node.m

Model class for every gameobject in our game. Takes care of rendering and transformations for gameobjects. Also takes care of applying textures to gameobjects. Node also has a field for children, which allows a gameobject derving from node to have other gameobjects as children.

PNode.m

Derives from Node and takes care of bullet3 physics. Gameobjects will derive from PNode instead of Node if they need bullet3 physics. Pnode takes care of creating rigidbodies for both convex and concave objects. Also takes care of setting an object's mass, restitution, and friction. Pnode also syncs the rigidbody with the drawn model, which applies to rotation and position.

GameNodes/

These are gameobjects which are drawn in game and can either derive from Node or PNode. The objects are created using the data from the imported models. Objects use the Node/Pnode initwithshader method to initialize. Gameobjects can be added to a scene as its children to be rendered in game.

Shaders

BaseEffect.m

Compiles and links the vertex and fragment shaders.

SimpleFragment.glsl

Fragment shader. Takes in values for position, colour, texcoord, and normal. Also applies the given texture, and uses phone lighting with ambient, diffuse, and specular.

SimpleVertex.glsl

Vertex shader. Takes in values for position, colour, texcoord, and normal.

Scenes

GameOver.mm

Scene which derives from Node and handles which game over text model to display.

GameScene.mm

Main scene where gameplay takes place. Gameobjects are added to its children and rendered. Handles game logic for tanks, collision, shooting, and turns.

General/Helper

Director.m

Singleton class which handles sound and music. Also stores data which needs to persist through each scene and viewcontroller.

Viewcontroller.m

View controller for the main gameplay. Gamescene and gameover scene. Updates the ui and sets the current scene in director.

HighscoreViewController.m

Viewcontroller for the highscores display. Handles operations with coredata for updating, setting, and populating the ui with scores.

StoreViewController.m

ViewController for the store. Ui for players to upgrade their tanks. Updates values through the director class.

MainMenuViewController.m

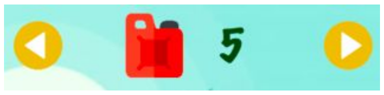
ViewController for the main menu. Initializes data for a new game. Has buttons for highscores and play game.

Game Design

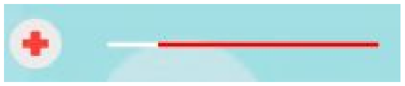
Level

Round-based

- Players alternate shooting
- Aims turret using touch
    - Swipe left/right to change angle
    - Swipe up/down to change power
- Launch projectile using on screen button
- Move tank using buttons on top middle of the screen



    - Gas jug icon and label represent movement limit
    - Gas limit refreshes every turn
- Player health indicated by plus icon and slider on their side of the screen



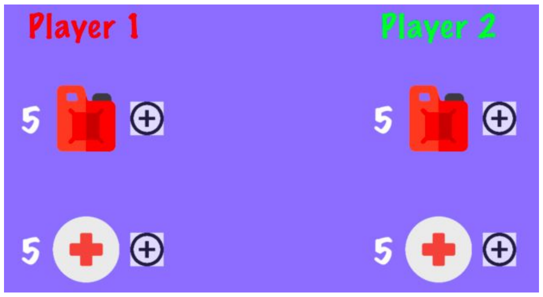
- Round ends when a player successfully destroys the opponent
- There are 3 rounds in total

Shop

At end of each round

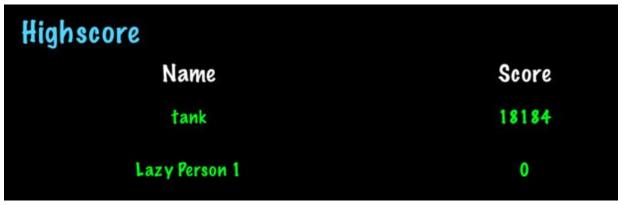
- Purchase health for more health

        - Purchase gas for more movement range



Score

- Player with highest score wins
- Score is calculated based on how fast a player launches their projectile
- Score is added to player opposite of the one that was hit by the projectile
    - A player can give score to the other by hitting himself
- Only the player with the highest score has the chance to post a new highscore
- A new highscore can only be set if it beats the lowest highscore already set



Highscore

Name	Score
tank	18184
Lazy Person 1	0

Player Profile

- Genre: Casual, Multiplayer, Arcade, Turn-based, Strategy
- Age: 4-32
- Gender: Neutral (skews to males because of subject matter)
- Mental Skill: Trajectory estimation
- Controls: One hand touch controls

Goals

Alpha

Met

- Basic Menu and UI
- Rendering
    - 2 tanks

    - Basic floor terrain
- Tanks can shoot and adjust the angle of their shot
- Basic turn based system
- Bullet Collision detection
- OBJ Loader
- Scene management and sounds

Unfinished

- Shop

Dropped from scope

- Deformable terrain
- Other projectile types

Beta

Met

- Shop
- Upgrade system - fuel and health
- Rounds and Turns
- Better Heads up Display
- Angle and Launch projection
- Bullet physics bugs fixed
    - Physics syncs with model when rotating for complicated rigidbodies
- Better Main Menu
- Game Over Screen
- Health and Damage System
- Game Loop
- Touch controls
- Core data setup

Final

Met

- Fix any bugs from Beta
- Scoring and leaderboards
- Main Menu Music and sound effects
- Fine-tuning features and aesthetics

- Optimization

Added

- Projectile trail line