Tom Agnew

Corey Stockton

CS437

Final Project

**GAME DESIGN DOCUMENT**

**Game Title:** Tankball!

**Game Concept:** Volleyball with tanks. Two players control opposing tanks that can shoot bullets and move back and forth. Players must shoot a ball with their bullets to knock it towards their opponents’ side of the playing field. If the ball lands on a player’s side, the opposing player gets a point and the ball is reset to the starting position. First to 5 points wins. Game window can be freely resized to alter the dimensions of the playing field, which changes the difficulty of the game.

**Game Objects:**

|  |
| --- |
| **Player** |
| Player-controlled tank. Can shoot bullets to hit the game ball. 2 players in a game, spawned on opposite sides of the game window facing towards each other. |
| Interactions:   * Bound behavior: STOP, cannot go past game window boundaries. Also cannot cross the halfway point of the game window. * Collision behavior (bullet): NONE * Collision behavior (ball): NONE |
| Controls:   * A/D or Left/Right (depending on player #): Move tank left or right. Cannot change direction tank is facing, so they are always shooting towards the opposing side. * W/S or Up/Down (depending on player #): Aim tank Turret (0 to 90 degrees). Rotates Turret object associated with Player. * Space or Right CTRL (depending on player #): Fire cannon |

|  |
| --- |
| **Turret** |
| The turret portion of the tank. Acts as a separate sprite so that it can rotate freely. Always associated with a Player object. |
| Interactions:   * Bound behavior: NONE, simply moves along with the Player object, so it always stays in bounds. * Sprites rotates along as Player adjusts aim. * Collision behavior (bullet): NONE * Collision behavior (ball): NONE |

|  |
| --- |
| **Ball** |
| Game ball. Hit back and forth by the players’ bullets attempting to get it to land on their opponents’ side. 1 ball in a game, spawned in the center of the game window midair. |
| Interactions:   * Bound behavior: BOUNCE, loses velocity on side walls. STOPs if it touches the bottom boundary (floor) and resets to starting position. * Collision behavior (bullet): Adds a force vector to the ball depending on where the bullet collides with it. * Collision behavior (player): None * Affected by gravity |

|  |
| --- |
| **Bullet** |
| Bullet shot by player tank. Can hit the ball to add a force vector to it, changing its course in the air. Spawns on the player when player fires cannon and flies in the direction the player is aiming their cannon. |
| Interactions:   * Bound behavior: DELETE (not actually deleted, only hidden to be reused as a future tank shot) * Collision behavior (ball): Adds a force vector to the ball depending on where the bullet collides with it. * Collision behavior (player): None * Not affected by gravity, travels in a straight line |

|  |
| --- |
| **Divider** |
| Provide visual marker that separates the two halves of the field. Does not actually act as a physical divider. |
| Interactions:   * Bound behavior: NONE (does not move) * Collision behavior: NONE |

**Game States:**

RUNNING: Game loop is proceeding as normal. Player tank controls are active, ball physics are running, and bullets are traveling. Game instructions are hidden. Switches to PAUSE state by pressing either the RETURN key or the P key.

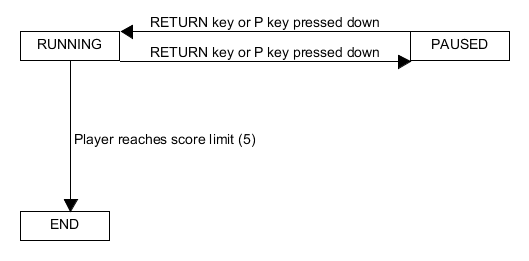
PAUSED: Game loop is paused in place. Player tank controls are disabled, ball physics are suspended, and traveling bullets are suspended. Game instructions are shown on screen. Switches to RUNNING state by pressing either the RETURN key or the P key. The game begins in the PAUSED state upon first being opened.

END: Game loop has finished. One of the players has reached the score limit and won the game while the game is RUNNING. Final results are shown on screen. Game can be reset to the starting state and conditions by pressing the R key.

**Game Scenes:**

Only one main scene where the tanks and ball are set up on the field. The current game score is shown at the top left of the game window as part of the UI. The scene is either in the PAUSED or RUNNING state during the course of the game and is in the END state at the end of the game.

**STATE TRANSITION DIAGRAM**

****

**SOFTWARE ENGINEERING PLAN**

**Primary Responsibilities:**

Tom Agnew is in charge of setting up the general game structure and gameplay. This includes:

* Creating central game loop, game states, and their transitions
* Creating Player, Ball, and Bullet objects, creating Player controls
* Creating game UI (scoreboard, instructions, etc.)
* Handling game scalability (properly adjusting to window resizes)

Corey Stockton is in charge of the physics interactions in the game. This includes:

* Coding how the Ball physics work (how forces are applied to the ball, how the ball interacts with gravity and window boundaries, etc.)
* Determining how Bullets are spawned
* Calculating Bullet trajectory through the air
* Calculating Bullet collision with the ball (when and where)

Though these are the primary responsibilities of the team members, the tasks are not completely exclusive from one another. Miscellaneous tasks and fixes are handled by both team members. Game design is a joint effort by both team members. Art and sound asset creation/collection is a joint effort by both team members.

**Project Timeline:**

**Proposed Timeline**

Nov 26 – Base game objects created, basic functionality in place. Game loop and states set up for further expansion.

Nov 29 – Player controls in place. Ball physics working, but no interaction with bullets yet.

Dec 3 – Game loop and state transitions finalized. Working UI in place.

Dec 5 – Bullets working and interacting with ball physics. Art and sound assets finalized.

**Actual Timeline**

Dec 3 – Base game objects created, basic functionality in place. Game loop and states set up for further expansion.

Dec 4 – Game loop and state transitions finalized. Working UI in place.

Dec 5 – Player controls in place. Ball physics working. Bullets working and interacting with ball physics. Art and sound assets finalized.

**USER INSTRUCTIONS**

**Requirements**

Python 3.x and Pygame are required to be installed to run the game. If the game is not working, update both Python and Pygame to the latest versions. The “Consolas” font is required to be installed on your system. This should be a default font for most OSes. If you do not have this font installed, the font TTF file is included in the project files and can be installed by double-clicking it.

**Installation/Running**

On the github page, click “Clone or download” and then “Download ZIP” to download the project zip file. Unzip the downloaded “cs437-final-project-master” folder. Double click “main.py” to run the game.

**Game Controls**

**General:**

Return/P – pause/unpause game

R – restart game (only after game is finished)

**Player 1:**

A/D – move tank left/right

W/S – adjust tank turret aim up/down

Space – fire bullet

**Player 2:**

Left/Right arrow keys – move tank left/right

Up/Down arrow keys – adjust tank turret aim up/down

Right Control – fire bullet