**Software Development for Games**

**Project 2 Final Document**

**Authors: Ben Mohan, Brian Kalvoda**

**Specification:**

* Story: You and any other players are tanks, you kill the enemy tanks and each other
* Scene: You and any other players are on a battlefield; you are ready to kill enemy tanks
* Players: 1-4 human/NPC players
* Objects: The tanks, the tank shells in flight, the ground
* Physics: Gravity on shell, velocity on shell, player tank stuck to ground, falling dirt
* Events: Player fires shell, player changes firing angle, player changes map position, player changes firing velocity, player gets hit by explosion, player dies, round ends, player buys ammunition
* Audio: Firing sound, impact sound, death music
* Rules: None, shoot at the enemy tanks
* Objectives: kill the enemy tanks before they kill you
* Platform: Web browser, mouse, keyboard if desired

**Initial Project 2 Step 2 Description:**

Our game begins with two or more tanks on a procedurally generated battlefield. It is a continuation of our tank game from project 1. Each tank is designed to be controlled by one person or NPC. Each player must adjust firing angle, position, and power to fire a shot that will impact the enemy tank. Players and NPCS take turns doing this until one remains standing. Our game also has terrain that is destroyed when exploded, forcing players to adapt to the changing terrain. Additionally, non-falling terrain will be implemented that will create interesting player opportunities and challenges. We will also be adding NPC’s (Non-playable characters) that will provide tanks on the battlefield. Another major change that we will be adding is multiple ammunition types that players will be able to purchase in between rounds. We will also be making changes to the visuals and size of the battlefield, to support more tanks on screen.

**Final Description:**

Our game begins with two to six tanks on a procedurally generated battlefield. It is a continuation of our tank game from Project 1. Each tank is a unique color and controlled by one person. Each player must adjust firing angle, position, and power to fire a shot that will impact the enemy tanks. Our game also has terrain that is destroyed when exploded, forcing players to adapt to the changing terrain. Also, multiple ammunition types have been implemented and can be purchased at the store with currency earned by damaging tanks. Finally, in addition to increasing the number of tanks, we increased the size of the playing field as well. Features we had planned to implement were non-falling terrain and NPC players.

**Design Notes:**

**Existing Features from Project 1:**

* Bitmaps: Both tank chassis are image files put into bitmap objects which are then manipulated in game
* Shapes: The ground objects and the tank objects, as well as all of the user controls are created from createJS shape objects and their graphics objects
* Animation
  + The tank shell shape is tweened through the sky and back down into the ground where it explodes by scaling up
* Mouse Input
  + All of the user controls are controllable through clicking on the on canvas shape buttons
* Keyboard Input
  + The firing angle is adjustable through the left and right arrow keys
* Containers
  + Each Tank is a container which contains a chassis and barrel which are then moved together throughout the playing field
* Sprites
  + The tank barrels are images that are put in their respective tank containers
* Sprite Animation
  + The tank barrels rotate based on the firing angle

**New Features Added in Project 2 Step 2:**

* Revised Specification (Ben and Brian)
* Tanks are implemented as self-contained objects now, allowing easier creation and manipulation of each one on the battlefield. (Brian)
* More tanks appear on screen, which is now easily implemented with the objects (Brian)
* Health now appears on screen always (Ben)
* A floating a marker appears above the player who is in control (Ben)
* Next turn begins after missile explodes (Brian)

**New Features Added for Project 2 Step 5:**

* Objects:
  + Missiles are implemented as objects with a parameter that determines the type (Ben)
* Menus:
  + A menu system is implemented that allows different amounts of players as well as the purchase of ammunition (Ben)
* Code Cleanup:
  + As part of tanks being made into objects, we also better encapsulated the functions that manipulate each tank and have fewer global variables. This will allow easier editing in the future. (Ben and Brian)
* Visuals:
  + New visuals allow more distinct tanks on screen each game and a more dynamic battlefield (Ben & Brian)

**Unimplemented Features:**

* Physics:
  + We plan on implementing non-falling terrain blocks that allow tanks to tunnel underground for new positions
* Objects:
  + Add AI controlled NPC’s

**Detailed Program Descriptions:**

* Missile: The Missile object contains all of the functions relating to the physics and states of the missile shape. Instances of this class are created when the Tank class’s getMissile function is called. Stage location of the missile is updated in the core game Tick function. Returns a shape object with attached functions.
* Tank: The Tank object contains all of the sprites and functions containing to Tanks. Functions included are the functions to move, change the angle of the barrel and power, and the missile types. The tank object returns a container that is then added to the stage in the core tanks class
* Button: The button object is simply a container class for a container object that acts as a button template. Allows entering of text to appear on the button and the x position in the box.

**Design Issues:**

* Creating new instances of a game after completing the first one proved to be quite challenging. When we removed all of the tickers from the stage, Tween no longer worked in the game.
* We decided to store the types of missiles in a single variable in Tank, rather than in the Missile class, so that each Tank had a unique inventory count for each missile type.
* We weren’t sure how many tanks we should allow on screen at once, but we tried different numbers until we settled on one that seemed right.