****

**Program by Wen Li**

**Maps by Kevin Xu, Cory Sulpizi, and Wen Li**

**ICS3U1 Mr.Wilford**

Wen’s Tanks is a computer based arcade game similar to Nintendo’s Battle City. It is a two-player game in which each player takes command of a tank with the goal of working together and taking out the endless enemy waves while protecting their home base. At the start of each level, the two tanks spawn beside the base protected by a layer of bricks. From this position, they will try to hit the enemy tanks that spawn periodically at the top of the map. There will be a set amount of enemies of different difficulties, and the players will have to adapt to the different maps and the different enemies in order to complete each level. As the players pass each level, the situation for them will be harder each time, but there will also be upgrades that spawn at random times to prepare the players for the higher levels. These upgrades will help the players kill the enemy tanks more efficiently and protect their base, and will be invaluable to them as the levels continue to go on.

**Rules**

When the game is opened, the player should move the tank to the desired command to start the game. The objective is to protect the base from bullets and to kill the enemies on each level that threaten both the players and the base.

The enemies will come in seven kinds, each with different characteristics:

* Normal: Moves and fires at the slowest pace and only has 1 hit point.
* Light: Moves at a fast pace, but fires quite slowly and only has 1 hit point.
* Upgraded Light: same as light, but with 2 hit points.
* Heavy: Moves quite slowly, but fires fast. They have 1 hit point each.
* Upgraded Heavy: same as heavy, but with 2 hit points.
* Advanced Heavy: same as heavy, but with 3 hit points.
* Elite Heavy: same as heavy, but with 4 hit points.

The colours of the enemies will indicate their health:

1 hit point – White, 2 hit points – Yellow, 3 hit points – Pink, 4 hit points – Red.

Another important aspect of the game to consider each level would be the different terrain types:

* Empty: Black squares that allow anything to come through.
* Grass: Squares that have vegetation; they allow tanks, but not bullets to be camouflaged.
* Water: Squares that allow bullets, but not tanks to pass through it.
* Brick: Squares that restrict bullets and tanks, but is destructible in two hits.
* Metal: Squares that restrict bullets and tanks; only 4-star player tanks may destroy them.
* Base: A square with a leaf that is placed at the bottom centre of each map. If this square is hit, the game will be over.

The map also has different upgrades a player can get:

* Star: Upgrade player to a maximum of four stars. If player is killed, a star will be taken away.
* Red Cross: Increase life of player by 1.
* Lightning: Make player temporarily invulnerable.
* Brick Frame: Fix the base of any empty or crumbling blocks.
* Metal Box: Make base temporarily invulnerable to enemy fire and fixes the base.

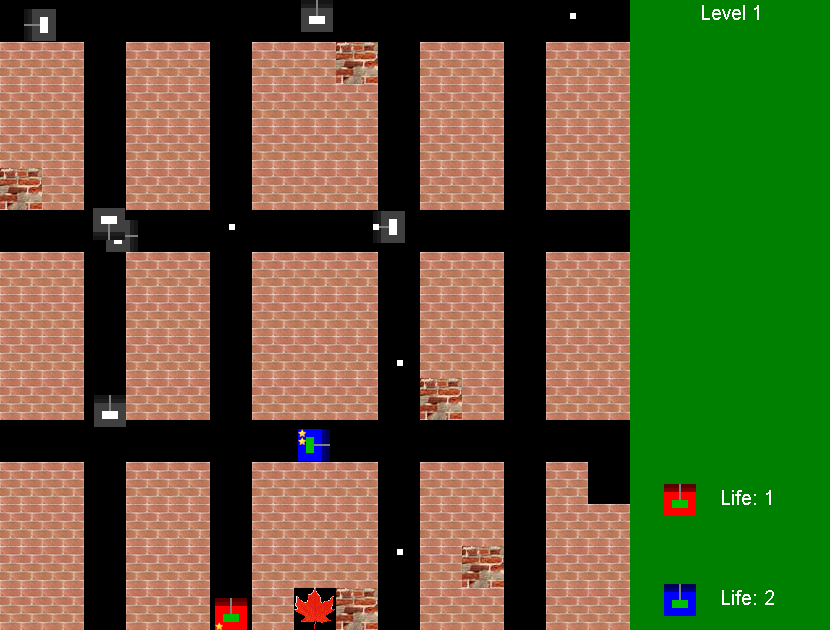
**Controls**

|  |  |  |
| --- | --- | --- |
| **Control** | **Player 1 (Red Tank)** | **Player 2 (Blue Tank)** |
| **Move Up** | W | ↑ (Arrow Keys) |
| **Move Down** | S | ↓ |
| **Move Left** | A | ← |
| **Move Right** | D | → |
| **Fire** | 1 (Top Row) | / (Forward Slash) |

Pause – Space Bar

**Screen Shots**

****

****

**Commentary**

**Process**

The process I went through of writing my program went in a series of steps that were completed on a daily basis:

1. Tanks drawn and moved.
2. Map drawn.
3. Tanks moved inside map correctly.
4. Bullets fired and registered.
5. Enemy tanks spawned.
6. Tank upgrades (different types of tanks).
7. Spawn animation, death animation, and inter-level screen.
8. Information screen.
9. Upgrades (power-ups).
10. Basic artificial intelligence.
11. Final artificial intelligence.

Over the next several days, the start screen, the victory screen, and minor bugs were fixed for the final program.

**Organization**

My final program had seven classes that were all interconnected in a chain like this:

**Problems & Solutions**

During the process of making my program, I encountered some difficulties and needed solutions for them. Below are some of the more difficult problems I faced and how I overcame them.

Tank

The first challenge I encountered was drawing and moving the tanks in the screen. Unlike the balls that we drew in class that moved the same way in all directions, the tanks had to face different directions if they were moved in different directions. I first experimented with different formulae, but in the end, with the help of a diagram, I was able to have the tank’s coordinates. By flipping around the diagram, I easily came up with the formulae for each part of the tank for whichever direction it is facing. From this problem, I learned that visual representations can simplify any complex problem.

(x, y)

Map

Another of the problems in the beginning was how I would make my map for the game. I could just draw a map of rectangles and squares, but I needed a way to do so that would allow me to create other maps and simply change them from level to level. After some thought, I thought I would create a 2-dimensional map of blocks across the screen. I would make a class called Terrain, and this would be an object that would contain the image and coordinates of a block. I would then create a 2-dimensional array of this object to create the map. This method worked out very well, and with the help of my friends, I was able to get many maps for my game. I learned that separating a 2-dimensional screen into these blocks really helps to simply the problem.

Artificial Intelligence

Of all of the problems I encountered, artificial intelligence was perhaps the one that took the most thinking. A human can easily decide what to do in many different kinds of situations, but the computer doesn’t know when to even move forward. After making my code as abstract as possible, I created scenarios for the enemy tanks, and they all depended upon whether or not the tank is able to move in a certain direction, which was determined by using methods that moved the tank and then checking if the position is valid. If the tank was in open space, I preferred it going forward. If the tank had a choice to go forward, left or right, I made it randomly choose a direction. If it can’t go in any of these three directions, I turned the tank back around. If, however, the tank entered a place that is near the base and is in a direct line with it, I made it so that there is a very high chance of the tank turning toward the base and shooting at it. Besides these systematic moves, I also added in a small chance for the tank to turn in a random direction, making them unpredictable. After tweaking the variables that controlled the chance of all of these happening, the end result turned out to be very desirable. The enemy tanks were both hard to predict and challenging, making the game fun to play and worthwhile.