

**GAME ALGORITHMS (TGD3351)**

**TRIMESTER 1 2020/2021**

**Milestone Report #2**

|  |  |
| --- | --- |
| **Student ID:**  1171101517 | **Student Name:** Michelle Chai Mei Wei |
| **Student ID:**  1171100973 | **Student Name:**  Foo Fang Jee |

# Work Done

1. Boss

We have implemented finite state machines on boss where the boss has 3 states which are avoid, attack, attack faster. Boss will enter avoid state when the player starts firing. Boss will move to left or right depends on the player position. Boss will enter attack state after player stops firing. In this state, boss will move towards to the player and fire towards the player if the player is within line of sight at fire rate of 150 milliseconds. Boss will enter attack faster state once the life is less than 30% and the boss is in attack state at fire rate of 75 milliseconds. Currently, boss will stay in avoid state if the player keeps firing, we plan to change the state to attack if the player fires for more than 10 seconds.

1. All the enemies – enemy\_1, enemy\_2, boss, turret

The line of sight algorithm for all enemies, except asteroid (since they do not fire bullets) are completed. It takes into account the distance from enemy to player, the angle distance (cone), and a Bresenham’s line check is performed to find any obstacles in between the enemy and player.

1. Missile

Player can shoot a missile with a cooldown of 10 seconds. We have implemented pathfinding for the missile’s path using A\* algorithm and Bresenham’s Line Algorithm.

1. Pattern movement

Implementation of a Catmull-spline curve is done, but the implementation of the curve to the positions of the enemies still needs more work.

1. Turret

Turret is an enemy type that does not have any movement algorithm. It has line of sight; the distance is 200.0f with an angle of 90.

1. Pause Scene

When the player presses the “Enter” key, the game is paused.

1. Collision

Previously, we used the AABB Collision Detection provided in Lab 1. Since we implemented a bounding box for each of our game object, collision is simplified through using Intersects method. Example is provided below:

if (player.BoundingBox.Intersects(enemyBulletList[i].BoundingBox))

# Upcoming Task

## Coding

1. Pattern Movement update

To have a list of control points so that the enemy will move in a spline curve

1. User Interface (UI)

Score, health, menu button, missile in the gameplay scene

1. Powerup

Either to provide invulnerability or to change the player’s bullet pattern

1. Level design
   1. Tutorial

The first level will be a super easy level, where we allow the player to learn the mechanics of our game, shoot (space bar), missile (z), move left, right, up, and down. We are hoping to make pop-ups to teach the player too.

## Documentation

1. Final Report

# Problem Encountered

## Pathfinding causes the game to lag

When we first implemented the pathfinding algorithm through A\* algorithm, we considered all the coordinates in the game scene, as in every single pixel, and this had caused A\* algorithm to have many possibilities in exploring the nodes.

Solution: Instead of adding a single pixel, we add 100 pixels when exploring neighbor node. This can reduce unnecessary nodes when exploring the neighbor. Besides, we also use Bresenham’s Line Algorithm in the pathfinding. If there are no obstacles along the path, we will use the result from Bresenham’s Line Algorithm. This can reduce the call of A\* algorithm, thus reduce the computing time.

## Catmull-spline curve

The enemies are currently moving in a specific angle correspondent to the single catmull-spline curve calculated. We believe it requires more control points but we are not very sure. So, more trial and error are required.

# Proposal Revision

1. **Pathfinding**

We intended to implement the pathfinding using A\* algorithm. However, A\* algorithm takes long computing time. To solve the problem, we combined the Bresenham’s Line Algorithm together with A\* algorithm.