**Group12**

Choi Ka Chon 1155051770

HUANG Chuanshan 1155076957

Siu Pong Yin 1155062979

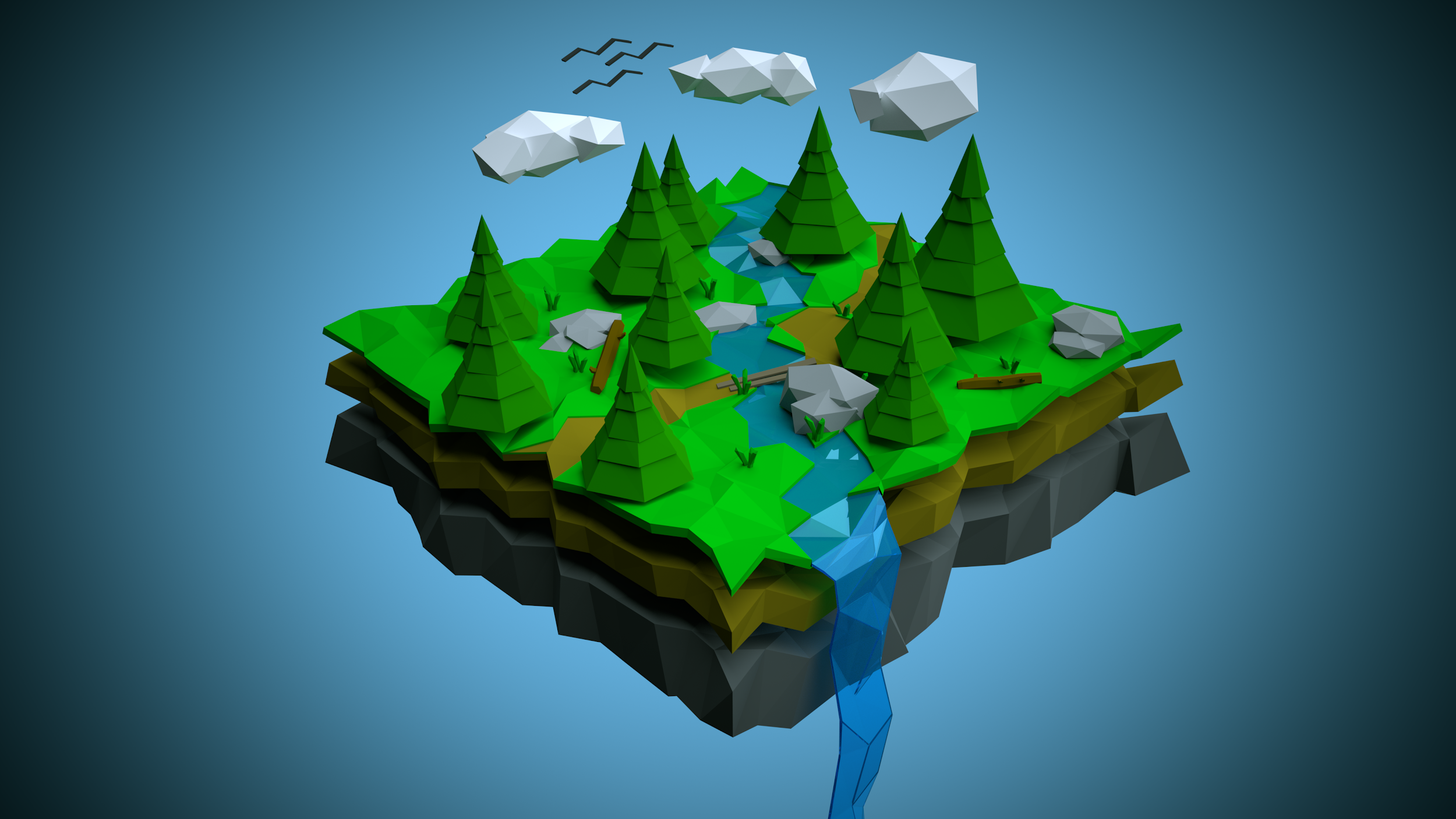
Wong Chak Shun 1155061036

**Premise of the game**

AntiSlime is a Low Poly shooting game where player can absorb different elements from the terrain to counter slimes of varying weaknesses so that they can survive the attack from the hostile slimes in an isolated island. Under usual circumstances, players spend around 10-15 minutes in our tailor-made environment to get through a level with specific goals.

**Intended Audience**

We adopt the “Lookalike Audience” method to define our target audience. We choose some similar games to AntiSlime, such as Monument Valley or any other third-person nonhardcore shooter game, look into their demographics and analyze their intersection and union. Finally we come a solution that our target audience might be general non-hardcore player who would like to spend some minutes to play our game to get pass their spare time, and those FPS players who also have a lust in survival game (since we have some survival elements).



**Genre**

This game will be a **3D single-player shooting game** in **Low Poly Style**, in which players will use a weapon (cannon) to convert absorbed elements into specific ammunitions for countering the monsters, a.k.a Slimes, in order to pass through levels.

**Overall Storytelling**

The main character is trapped in an isolated island covering with slimes. They are resentful with this newcomer. He must fight against the slimes to survive in the island. Fortunately, he is equipped a powerful cannon which can turn different elements into firing ammo. Players should use their intelligence to help our main character overcome the attacks from slimes and escape safely.

**Game Design**

Players will play this in the **third-person perspective or first-person perspective** camera. This can be switched between by players in-game. About game structure, Antislime will adopt the **semi-linear** style, where players have an explicit goal in each level, for example: survive from the slimes’ attacks, kill all the slimes or reach the escape points. Nevertheless, players can also spend time on collecting different elements. They have to manage their usage of certain elements in order to kill particular slimes. Therefore, they have to traverse the map carefully to collect elements and try hard to stay away from slimes as once players got killed they need to restart the level.

**Game Play**

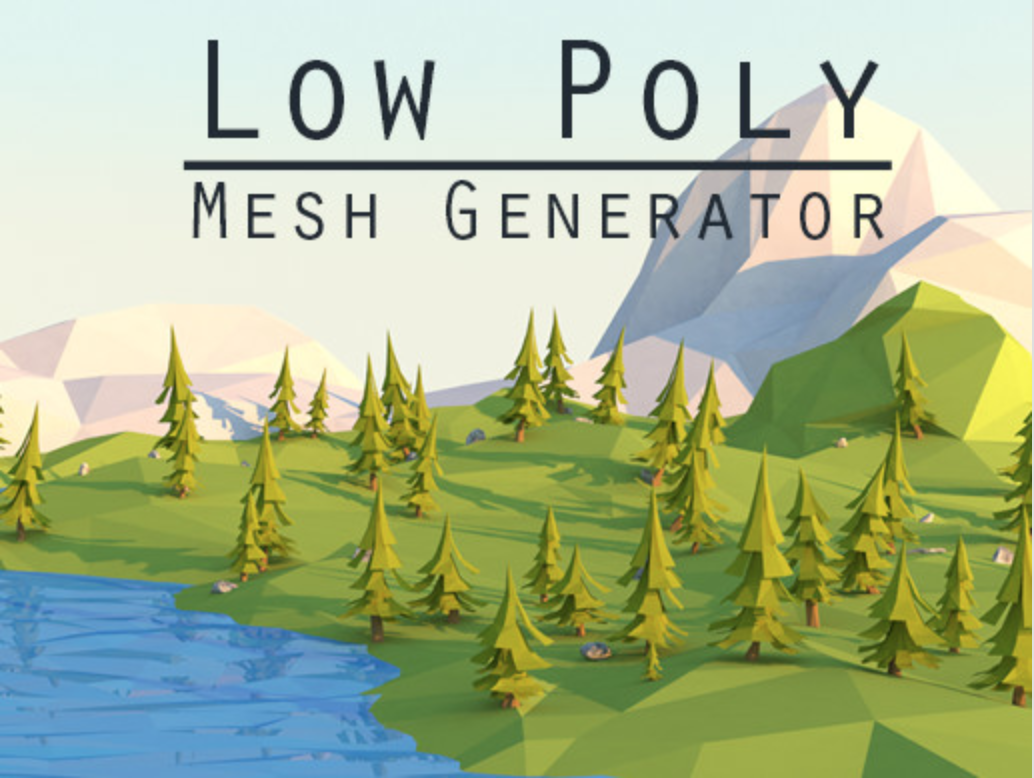
The main character is able to **collect the five existing elements (fire, water, wood, earth and metal)**, scattering in the map of each level. The collected elements will be converted into certain amount of ammunition automatically. Players can fire to anywhere, with any elements in their storage. However, **some elements may be more effective in killing certain types of slimes.** For example, the water element can apply hugest damage on fire slimes. So players should choose the elements carefully. Except firing at slimes, players can use elements to build up barriers to block slimes from attacking the main character.

**Technology & Resources**

Two main core technologies, **dynamic pathfinding** and **spawning of enemies**, will be used in our game. In order to provide the best experience for our players, enemy AI is extremely important in the gameplay. The enemies should chase the player once they saw him and can find out the optimal path to the player in the dynamically changing world. Spawning of enemies also play an important role in the gameplay experience. The enemies should be evenly distributed in the game world and balance itself by spawning new enemies in the right spawning point, and the position of spawning points are very important as well.

**Blood shader** is also one of our game feature. When the player is hurt or in low blood level it will trigger a bleeding effect, which the screen might turn to grey scale or bleeding at the four corners of the screen.

In case of modeling and environment setup, we plan to take advantage of Unity Asset Store, where **free Low Poly packages** are available (so that we can build the map directly with existing models), or we can use the **Low Poly Mesh Generator** to convert our High Poly 3D terrain into a Low Poly one.



Below are the pseudo-code illustrate the high-level concept of dynamic pathfinding and the spawning of enemies.

***Dynamic pathfinding:***

Function A\*():

closeSet = []

openSet = []

openSet.append(start)

while(true):

current = node in open with the lowest fCost

openSet.remove(current)

closeSet.add(current)

if current is target node:

return

for neighbour in currentNodeNeighbour:

if neighbour is traversable or neighbour in closeSet:

continue

if new path is shorter or neighbour not in openSet:

calculate fCost

parent = current

if neighbour not in openSet

openSet.append(neighbour)

// On every frame update

Function mainLoop:

// Enemies see the player

If fieldOfView is true:

// Update the map

mapUpdate()

// Find the path

gameObject.position = retrivePath(A\*()).next()

***Spawning:***

Function mainLoop:

// set a routine in certain time interval

rotine(function spawn, time spawnTime)

Function spawn:

If playerHealth != 0 and balance():

// choice spawn point

// random function can replaced by other optimise algorithm

spawnPoint = Random(vec2[] spawnPointList)

createEnemy(Object enemy, vec2 spawnPoint)

Function balance:

// This condition can be improved by saving the last enemies

// killed and calculate the possibilities of the next enemies

// killed

If numEnemies > maxEnemies:

return falase

else:

return true

~ END ~