2017

LYU 1702: Augmented Reality Game with Tango

TANG DING 1155047074
TANG XUN 1155046996

Contents

1. Abstract

Although AR shows great prospects, there’s no existing application which can exploit its potential. There’s many reasons for such situation, and the most important one is that developers haven’t fully realized what AR can do and how it should be used.

Since Google Tango is a convenient and not so expensive, it is a good choice for use to explore the AR world.

In the first semester, we did a research into Tango’s mechanism and functionalities and developed an AR mobile game in order to make a better understanding of Tango development. Also, we build a basis for future multiplayer mode and AR educational platform.

2. Background

Virtual reality is becoming more and more popular, since hardware performance has been able to make virtual images often indistinguishable from the real world. However, those images presented in computer games, science fiction films, or other media are disconnected with our physical surroundings.

Smartphones and tablets provide a convenient way to gain huge size of information. But it is usually detached from the physical world. Consumers have to do a lot to connect such information with world around us. There 're many attractive ways that try to connect the mobile computation with real world, such as GPS or barcode. But their applicable fields is too few and cannot handle emergency.

Augmented reality (AR) is thought to be able to build direct, automatic, and actionable connections between the real world and virtual information.

Definition and Scope

AR is intended to connect virtual world with real world, both spatially and cognitively. With AR, the virtual information appears to become part of the real world. Building this connection is a great goal—one that requires knowledge from many fields of computer science, yet can result in misunderstandings about what

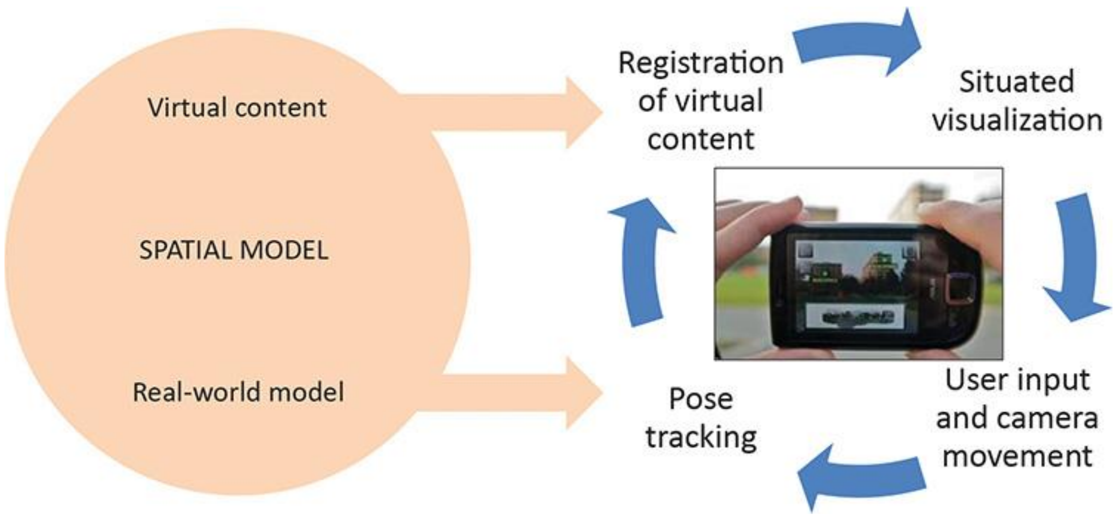
AR really is. So, let’s answer this key question: What is AR?

The most widely accepted definition of AR was proposed by Azuma in 1997 [1]. According to Azuma, the following three characteristics are necessary for AR:

- Combines real and virtual
- Interactive in real time
- Registered in 3D

Note that the definition does require real-time control and spatial registration, meaning accurate real-time match of corresponding virtual and real information. Although ideas on the standard of real-time performance may be different according to the different people and different tasks, interactivity implies that the human–computer interface operates in a tightly coupled feedback loop. The user continuously navigates the AR scene and controls the AR experience. The system, in turn, picks up the user’s input by tracking the user’s viewpoint or pose. It registers the pose in the real world with the virtual content, and then presents to the user a situated visualization [2].

A complete AR system requires at least three components: a tracking component, a registration component, and a visualization component. Usually a database or other storage system is used to store data of the real and virtual world. And data stored is called spatial model. The real-world data is a referred to get the user's real position. The virtual-world data stores the content used for the augmentation.



The main AR platforms today include Google Tango, Microsoft HoloLens and Apple ARKit. In this project, we will focus on Tango.

3. Research in Tango

3.1. Overview

Tango is a platform that uses computer vision and other techniques to offer electronic devices the ability to understand the world surrounding them. It is achieved by using three core technologies: Motion Tracking, Area Learning, and Depth Perception [3].

- Motion Tracking overview

Motion Tracking implies that a Tango-enabled device can get its own trajectory in 3D world. Walk with a Tango device and rotate it to different directions, it can tell you its position and rotation.

- Area Learning overview

People recognize where they are by observing the features surrounding them: a corner, a wall, or a table. Tango can do similar thing.

With Area Learning turned on, the device can record the features it sees, and it can also store and load the data again. So a device can recognize its position with previous “memory” by seeing features that have seen before.

- Depth Perception overview

With depth perception, your device can understand the shape of your surroundings. This ensures that virtual things can interact with real world.

A Tango-enabled device is an Android device with a wide-angle camera, a depth sensing camera, accurate sensor timestamping, and a software stack that enables all Tango features.

3.2. Motion Tracking

Motion Tracking in Tango is achieved through **visual-inertial odometry**. It uses camera images and inertial motion sensors to estimate both its position and rotation in 3D world more accurately [4].

Pose

A device's **pose** means the combination its position and orientation of the user's device in full six degrees of freedom. There're two methods in Tango API to get pose data: one is using callbacks to get the most recent pose updates, and another is using functions to get a pose estimate at a specific time. The data consists of two main parts: a vector in meters for translation and a quaternion for rotation.

Common use cases

- **Improved rotation sensing**: It can replace Android Game Rotation Vector APIs with better performance.
- **Tracking movement**: Tango allows you to track a device's movement in the real world.
- **Virtual camera**: When you combine rotation and position tracking, you can use the device as a virtual camera in a 3D rendered environment.

Limitations

Motion Tracking is good but it does have several limitations:

- Through motion tracking, device does not truly understand the real world around.
- Every time you start a new Motion Tracking session, it will forget the previous information and cannot tell you where you are now.
- Over long distances and periods of time. Small errors will be accumulated to a large amount.

3.3. Area Learning

How it works

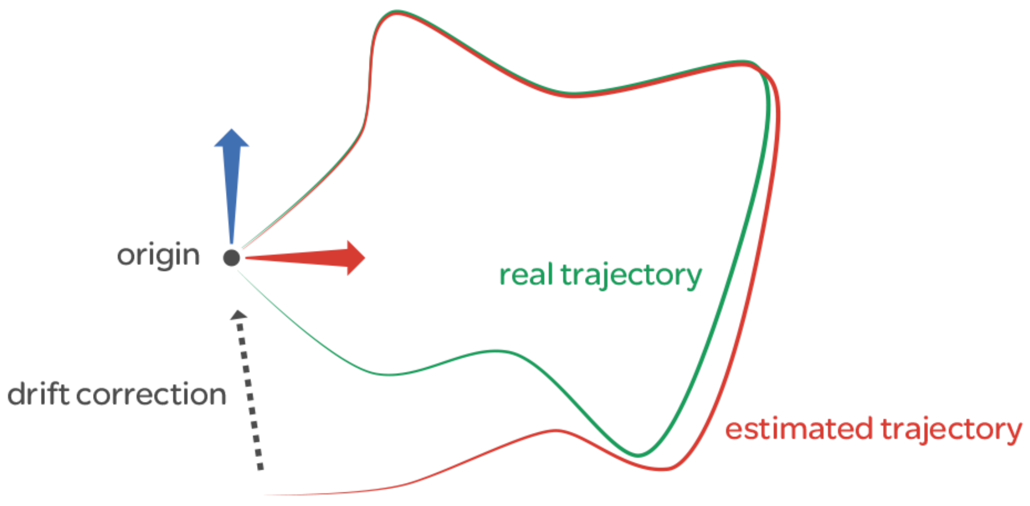
With Motion Tracking alone, device cannot keep memory of things it sees. But Area Learning ensures that it is able to “remember”. To do this, it generates a mathematical description of the edges, corners, other unique visual features [5].

When Tango has learned an area, it is able do two things to improve Motion Tracking:

1. **Improve the accuracy of the trajectory** by performing "drift corrections."
2. **Orient and position itself within a previously learned area** by performing "localization."

Improving the trajectory

As mentioned before, Motion tracking become less accurate over long distance and period of time. With Area Learning, the device is able to use the visual features in its memory to adjust its estimation on movement. This memory allows the system to perform **drift corrections**. When the device recognizes a visual feature it stored before, it knows a loop is finished and adjusts its trajectory to be more consistent with previous information.



Area descriptions and localization

As we mentioned before, Area Description generates a mathematical description of the edges, corners, other unique visual features. You can save this information into Area Description File (ADF) for future usage.

Common use cases

- **Multi-player experiences**: ADF is sharable. This allows multiple people to interact in the same physical space.

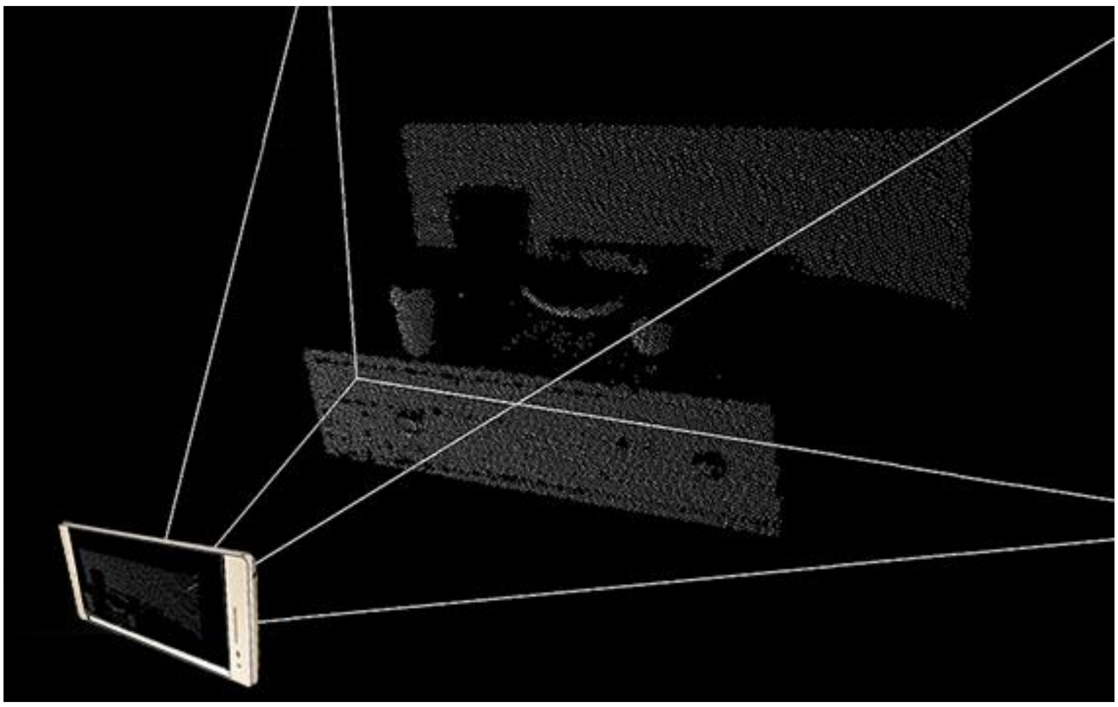
3.4. Depth Perception

How it works

Depth Perception provides a device the ability to estimate the distance to objects. The way of implementation is depended on manufacturers. They can choose among several depth technologies, including Structured Light, Time of Flight, and Stereo [6].

Point clouds

The Tango APIs provide a function to get depth data in the form of a point cloud. This format gives (x, y, z) coordinates for as many points in the scene as are possible to calculate. Each dimension is a floating point value recording the position of each point in meters in the coordinate frame of the depth-sensing camera.

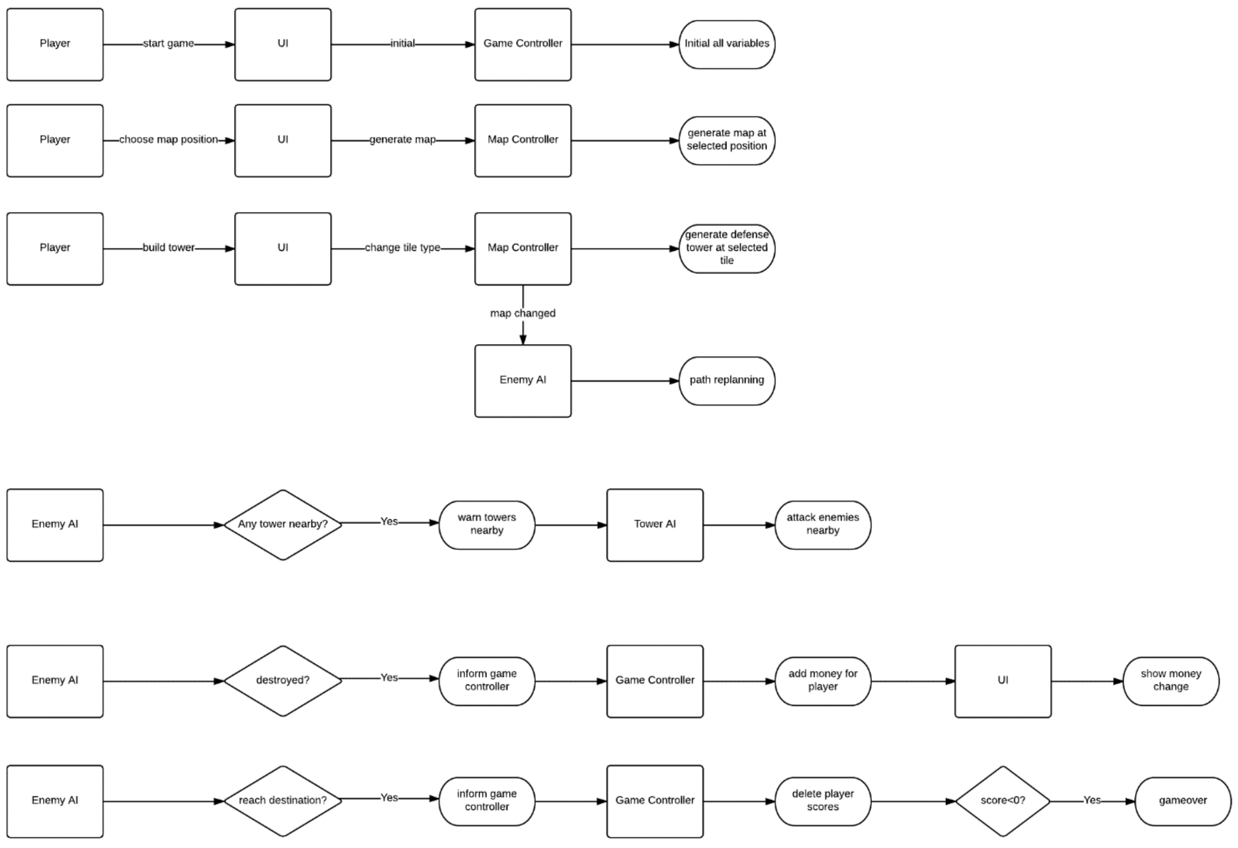


4. Application Design

In the beginning, we decided to develop a multiplayer AR game. However, since only one Tango device is provided, we are unable to develop and test multiplayer mode. So we changed our plan in halfway.

We designed a tower defense game, where players build different types of towers to prevent the enemies from reaching the terminal.

The basic logic flow of game is showed below:



The game can be divided into two phases:

- Preparation Phase

In this phase, player needs to start app and click screen to choose a position to begin game, and the position is required to be on a flat plane for convenience. The app will use Tango's depth perception feature to detect the plane, and generate a random maze-like map on this plane.

- Game Phase

In this phase, player needs to observe the map and enemies, and choose the best position to build defense towers. Those defense towers will automatically attack enemies nearby, and when an enemy is destroyed, player can gain money to build more towers. But if some enemies are not destroyed and reach the destination, the score of player will be reduced. When score of player is reduce to 0 or no more enemies are left, the game is over.

5. Application Implementation

5.1. Map Generation

We want to provide players with a unique and enjoyable experience every time. The final goal is to utilize the area description of Tango to build the map by reconstructing the environment. Currently we have succeeded in find the planes in the surroundings. When the player holds the device, and touches some plane the screen shows, such as a desk surface or a wall, we will generate the map terrain at that surface.

Currently we have managed to make the map generation randomized to make sure it will get different challenges each time. The map size is about 1 meter * 1 meter with no more than 21 * 21 blocks of floors or obstacles. It is reasonable to have some rooms and some narrow paths.

We adopt a Depth-First-Search(DFS) algorithm to generate the map with fixed start and end point and no dead ends.

5.2. Tower Design

In our game, the tower can be placed on the floor or on the obstacle. If be placed at the floor, the enemies' path finding will be affected to search a new path. If there's no path to reach the terminal, the enemy will keep the previous route and attacks the tower when encounters. Therefore, the tower has its health value. Towers also have their own attack range, within which the targets that closer to the terminal will be chosen as the targets with higher priority.

To make the game has more playability, we've design 4 types of towers with different functionalities (different attack methods), and each type has 3 different levels of different power, health and price to build.

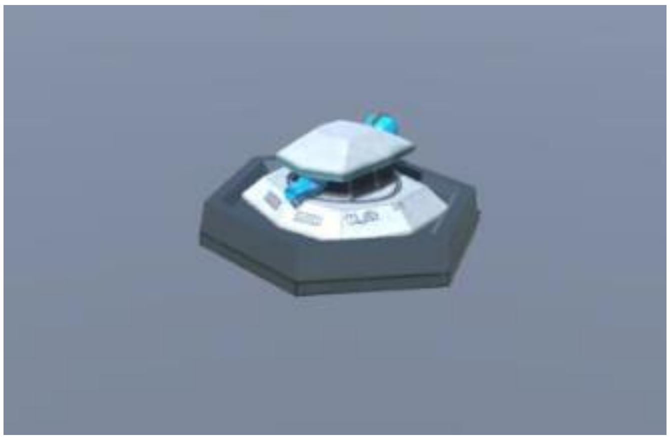
We get the tower models from the Asset Store of Unity.

5.2.1. Basic Towers

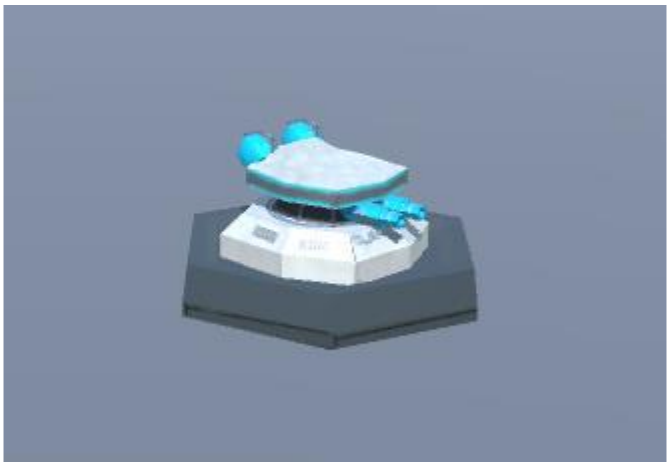
The Basic Towers rotate along Y-axis to face the target. The shoot shells to attack. We have implemented a smooth target aiming with prediction for the rotation along Y-axis, which makes no instantaneous direction turning of the turrets, also no sudden change in angular speed. Thus, the rotation looks physically reasonable and smooth. To implement this, in every frame, we calculated the relative angular "distance" and angular speed taking the targeted enemy as reference, based on which we figure out the angular acceleration for current frame to update the angular speed itself to reach our final state in the shortest time interval. In the final state, what the turret is going to achieve is that the angular “distance” is 0 (facing the target) and angular rotation speed keeps synchronous with the target moving. What’s more, as it needs some time for the shell to fly to the target, so the turret still need to rotate ahead of the target to shoot its shell. Hence, we calculate the distance between the turret and the target, with the velocity of the target, we can get the prediction position where the shell will hit the target.

The shells they shoot will explode when it hits the target or other objects. We have made the particle system for this explosion. There will be blast waves when they explode, which also have direction along the hit surface’s normal vector.

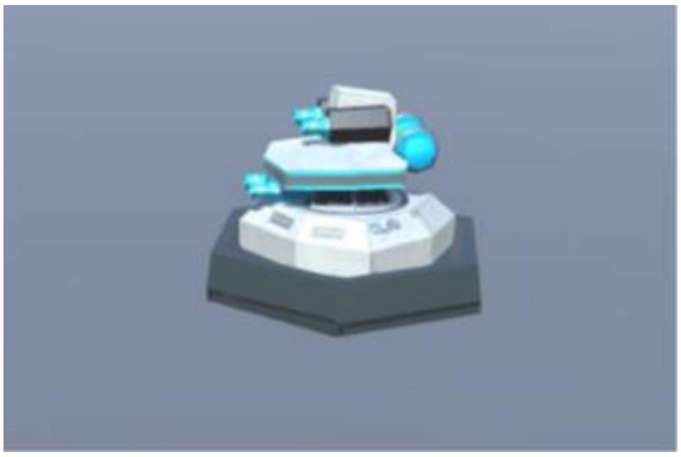
- Level1



- Level2



- Level3



5.2.2. Laser Towers

The Laser Towers rotate along Y-axis and X-axis to face the target. They emit laser beams to attack.

The rotation along Y-axis are the same with Basic Tower just without prediction, that is, it will not rotate ahead of the target but just rotate to face the target.

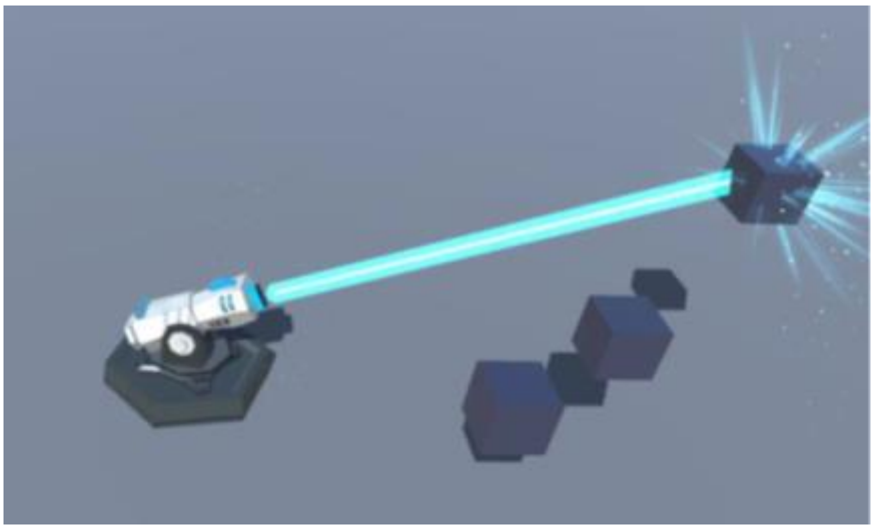
The rotation along X-axis will not have instantaneous movement as well.

We added angular restriction along X-axis to prevent from 3D model penetration. Thus, only the enemies within this angle range will be set as target.

We use particle systems to make the animations of energy charge, the blast waves and the energy burst when the laser is emitting, so that the effects will be more attractive.

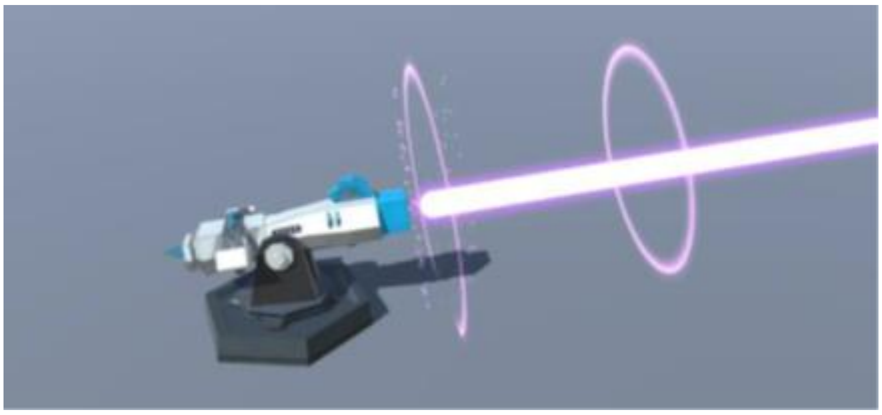
There we encountered a problem of the laser ray rendering. The ray somehow will be overlapped by the objects behind it. This is quite strange and we are going to figure this out in our further progress.

- Level1



- Level2

- Level3



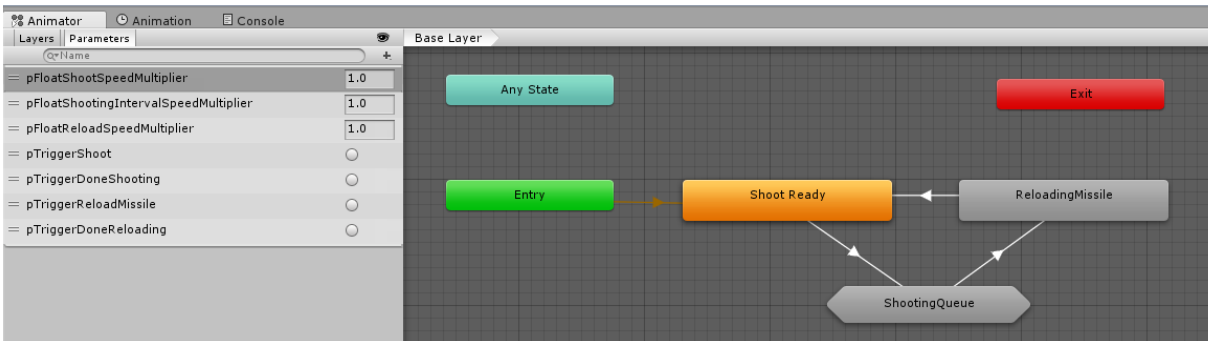
5.2.3. Rocket Towers

The Rocket Towers rotate along Y-axis and X-axis to face the target. They launch guided missiles to attack.

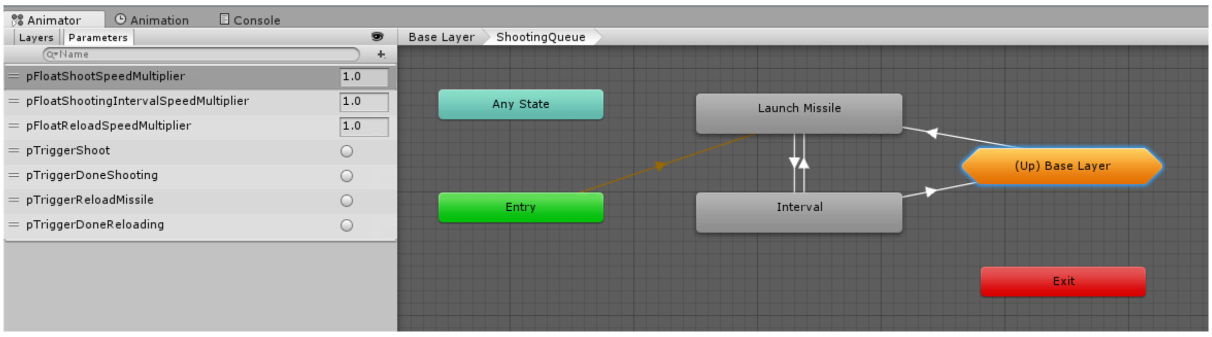
The rotation is similar as the Laser Tower.

We also add the animation to make it has recoil force while launching the missiles, so that the towers look more realistic. We use animator controller to control the sequences of launching missiles.

The states of the animator controller are shown in the below pictures. By setting the transfer conditions and the triggers, we control the tower to play the shooting animation. When the tower is ready to shoot and aiming the target, its state will transfer into the Sub-State Machine “ShootingQueue”, inside which, the state transfers between “Launch Missile” and “Interval” until all the missiles are launched. After that, the tower will go to state “ReloadingMissile” and then “Shoot Ready” to wait for next shoot request.

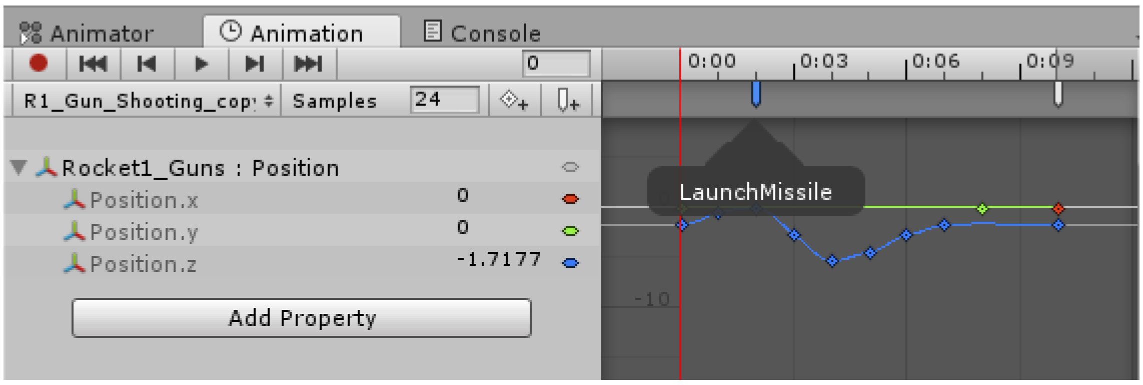


pFloatShootSpeedMultiplier	1.0
pFloatShootingIntervalSpeedMultiplier	1.0
pFloatReloadSpeedMultiplier	1.0
pTriggerShoot	○
pTriggerDoneShooting	○
pTriggerReloadMissile	○
pTriggerDoneReloading	○



pFloatShootSpeedMultiplier	1.0
pFloatShootingIntervalSpeedMultiplier	1.0
pFloatReloadSpeedMultiplier	1.0
pTriggerShoot	○
pTriggerDoneShooting	○
pTriggerReloadMissile	○
pTriggerDoneReloading	○

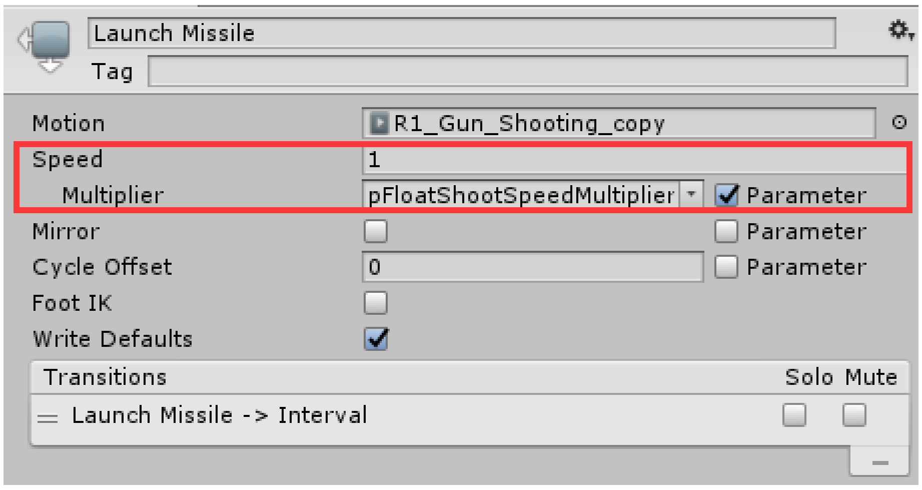
And through the animation events, we fire the missile launch method in our script to create a new missile object in the scene, so that we can make the launch of the missile at the right time when the recoil finishes.



Rocket1_Guns : Position

Position.x	0
Position.y	0
Position.z	-1.7177

We also managed to enable dynamically change of the animation duration. The state with animation enables setting the play speed with a multiplier of a parameter. And the value of the parameter can be changed during runtime.And therefore, we can define different duration for different levels of the tower through script.



Launch Missile
Tag

Motion	R1_Gun_Shooting_copy	
Speed	1	
Multiplier	pFloatShootSpeedMultiplier	✓ Parameter
Mirror	☐	☐ Parameter
Cycle Offset	0	☐ Parameter
Foot IK	☐	
Write Defaults	✓	

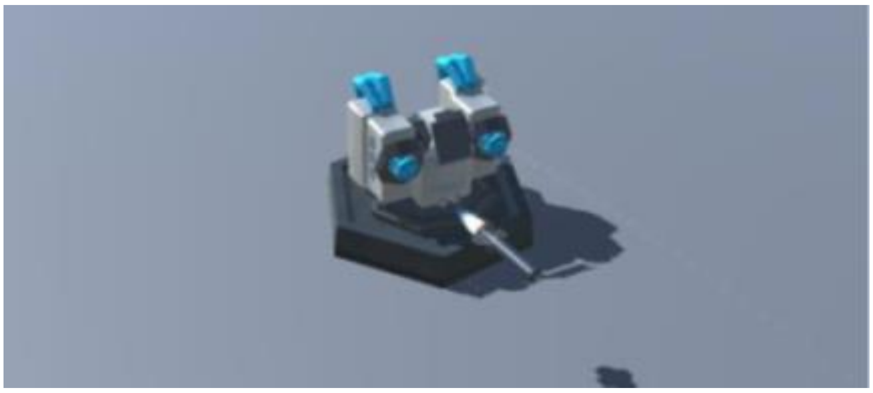
Transitions	Solo	Mute
Launch Missile -> Interval	☐	☐

The missile's movement is also physically reasonable. It flies forward and turns in small angular speed. Its maximum angular speed is inversely proportional to its speed, so when the target is not in just forward of it, it will decelerate to increase its angular speed to turn. After facing the target, it will then accelerate.

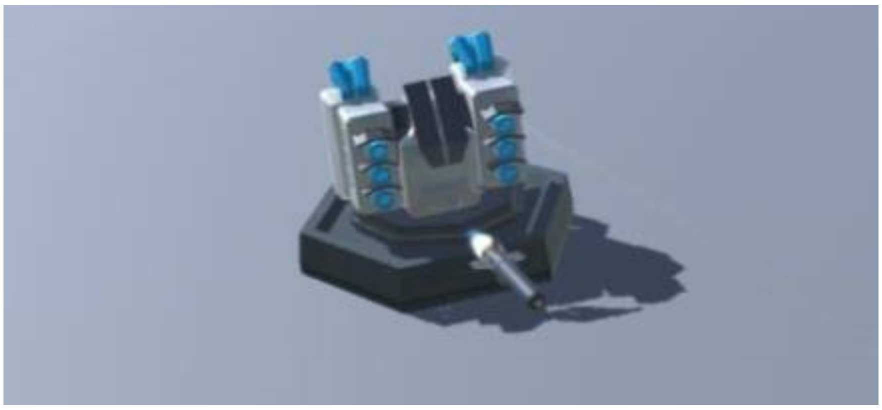
We use particle systems to form the fire and the smoke after the missile. The smoke will last for a while to compose the trajectory.

When the missile hit the target or other objects, it will explode. The particle system of the explosion is like the shell's one of Basic Tower.

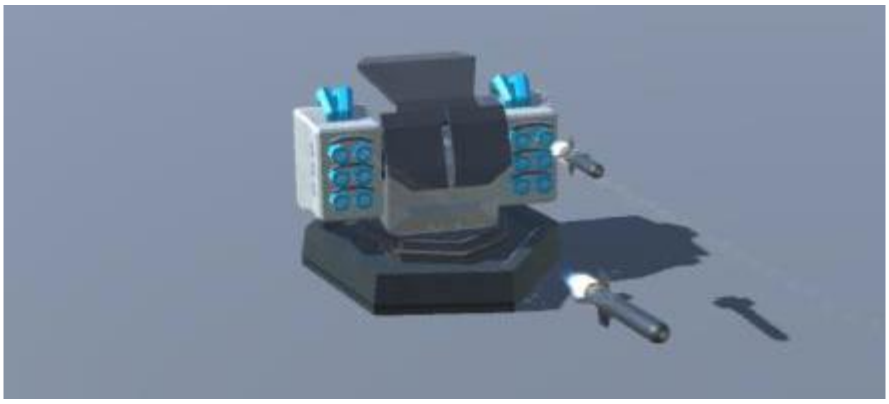
- Level1



- Level2



- Level3

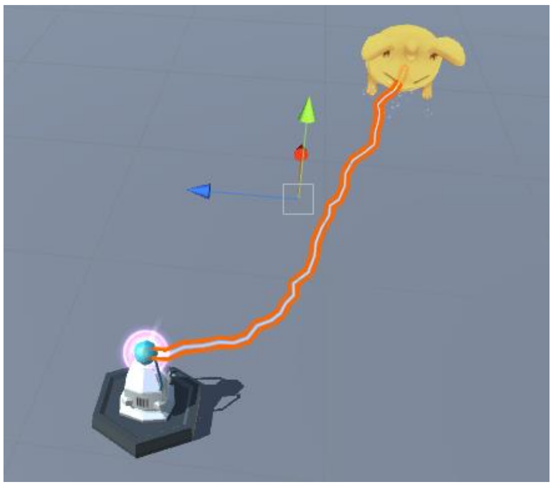


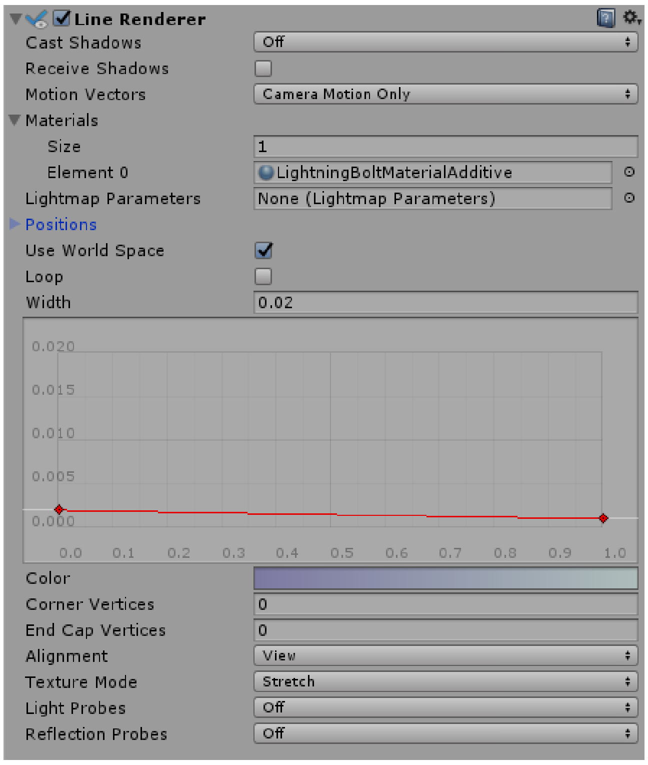
5.2.4. Tesla Towers

The Tesla Towers use electrical arc to attack multiple enemies. They will not rotate, but will show a ball lightning at the poles where the electrical arcs start.

This type of tower is different from the above ones that it can attack multiple enemies (has maximum) at one time.

For the electrical arc, it is created by Line Renderer. The Line Renderer of Unity is a powerful component to generate polyline. The “Positions” property of this component contains dozens of 3D points in x, y, z coordinates. We use program to randomly and reasonably update the points every several frames to make it behave like a lightning.



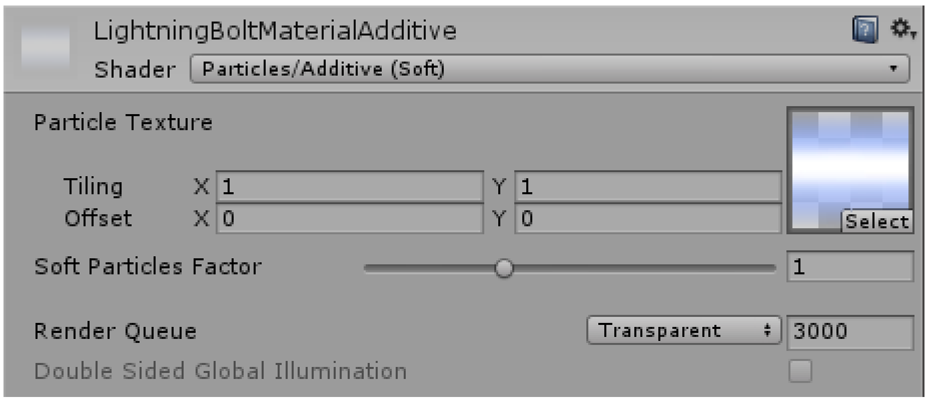


Line Renderer

Cast Shadows	Off
Receive Shadows	☐
Motion Vectors	Camera Motion Only
Materials	
Size	1
Element 0	LightningBoltMaterialAdditive
Lightmap Parameters	None (Lightmap Parameters)
Positions	
Use World Space	✓
Loop	☐
Width	0.02
Color	
Corner Vertices	0
End Cap Vertices	0
Alignment	View
Texture Mode	Stretch
Light Probes	Off
Reflection Probes	Off

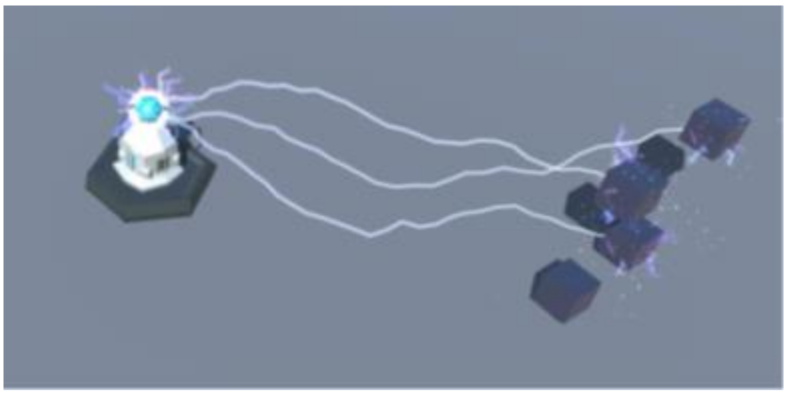
Though all the points are generated randomly, we need to confirm the start position and end position of this electrical arc, and the adjacent points should not be too apart from each other.

What's more, the lightning's material, width and color is also contributed by the Line Renderer, so we can manage different levels of Tesla Tower have different lightning effects.

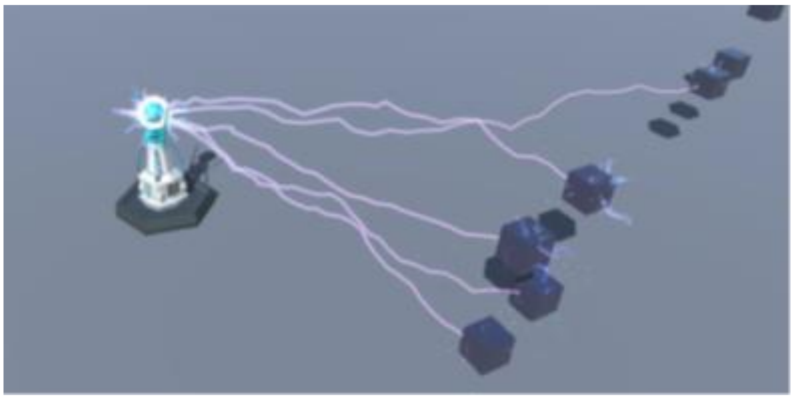


LightningBoltMaterialAdditive
Shader Particles/Additive (Soft)
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue Transparent 3000
Double Sided Global Illumination

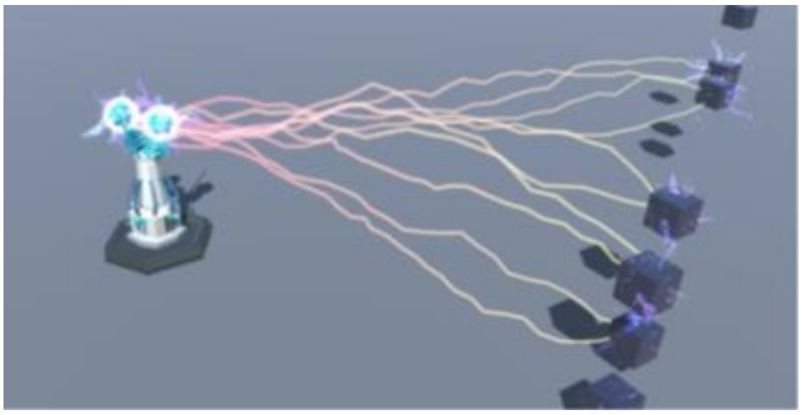
- Level1



- Level2



- Level3



5.3. Particle Systems

To add more attractive effects and beautiful graphics, we have used many Particle Systems. There are many phases of Particle System to control different mathematical transitions during its Lifetime. Main and Renderer phases are the most fundamental ones. The Main phase sets the initial properties of the particles; the Renderer phase controls the rendering methods and the materials.

In the Shape phase, it sets the distribution shape of the particles while generation. For example, a sphere or a circle.

Particle System allows using curves to control the size changing, gradients to control the color changing over lifetime in corresponding phases.

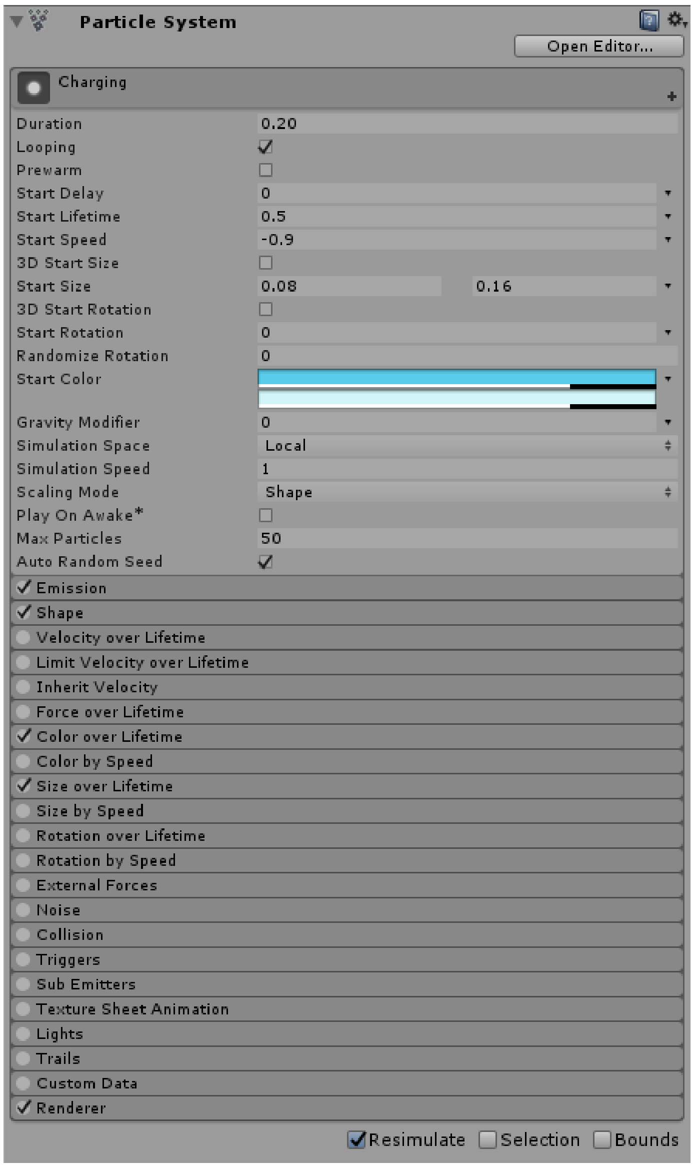
During our construction of these particle systems, we have encountered 2 problems.

➢ The scale mode of the particle system

Because the particle system in Unity cannot directly scale up and down to look similar, so the scale mode does not help but is troublesome. There are 3 types of scale mode: shape, local, and hierarchy. Even the scale mode is selected to hierarchy, most features will not follow the global scale of this particle system's transform. Almost only the size of the generation shape will scale. So we still need to manually change the size, speed and the gravity factor.

➢ The maximum particle size

This value is ambiguous and influence the foreshortening effect brought by perspective camera. When this value is not equal to 2, the particle may get larger when the camera moves further.



Particle System

Open Editor...

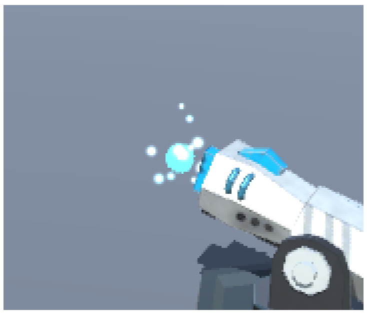
Charging

Duration	0.20
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	0.5
Start Speed	-0.9
3D Start Size	☐
Start Size	0.08 0.16
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Scaling Mode	Shape
Play On Awake*	☐
Max Particles	50
Auto Random Seed	✓

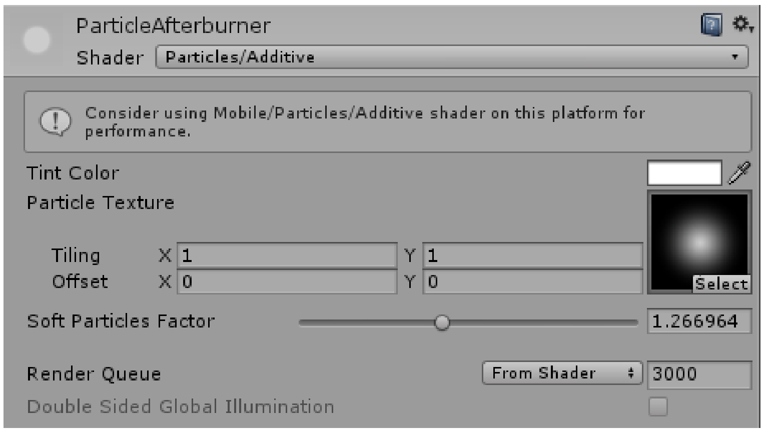
- ✓ Emission
- ✓ Shape
- Velocity over Lifetime
- Limit Velocity over Lifetime
- Inherit Velocity
- Force over Lifetime
- ✓ Color over Lifetime
- Color by Speed
- ✓ Size over Lifetime
- Size by Speed
- Rotation over Lifetime
- Rotation by Speed
- External Forces
- Noise
- Collision
- Triggers
- Sub Emitters
- Texture Sheet Animation
- Lights
- Trails
- Custom Data
- ✓ Renderer

✓ Resimulate ☐ Selection ☐ Bounds

5.3.1. Charging of Rocket Tower



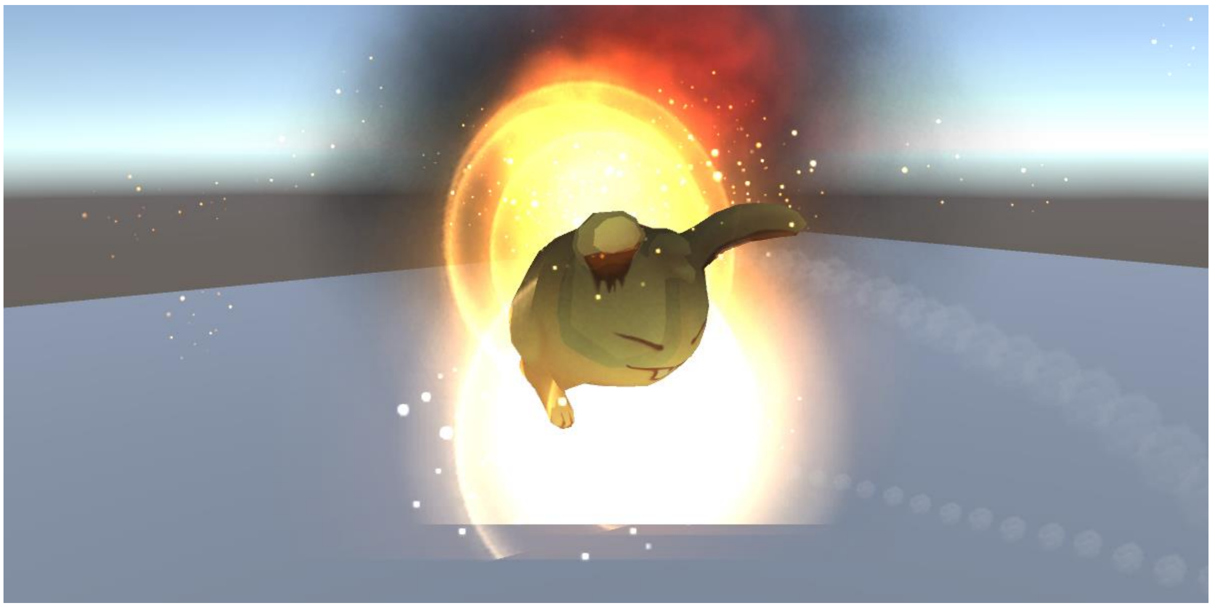
This particle system uses its material to generate particles uniformly distributed in a sphere surface and moving towards the center of the sphere. And over each particle’s lifetime, it changes its color and increase its size from invisible, and finally vanishes at the center.


ParticleAfterburner
Shader Particles/Additive

Consider using Mobile/Particles/Additive shader on this platform for performance.

Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1.266964
Render Queue From Shader 3000
Double Sided Global Illumination

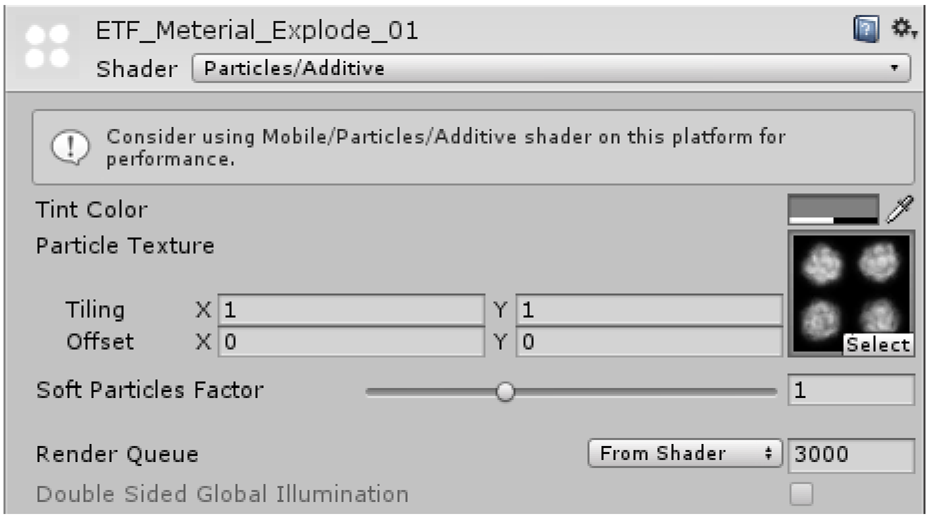
5.3.2. Shells and missiles Explosion



This explosion effect consists of 5 particle systems:

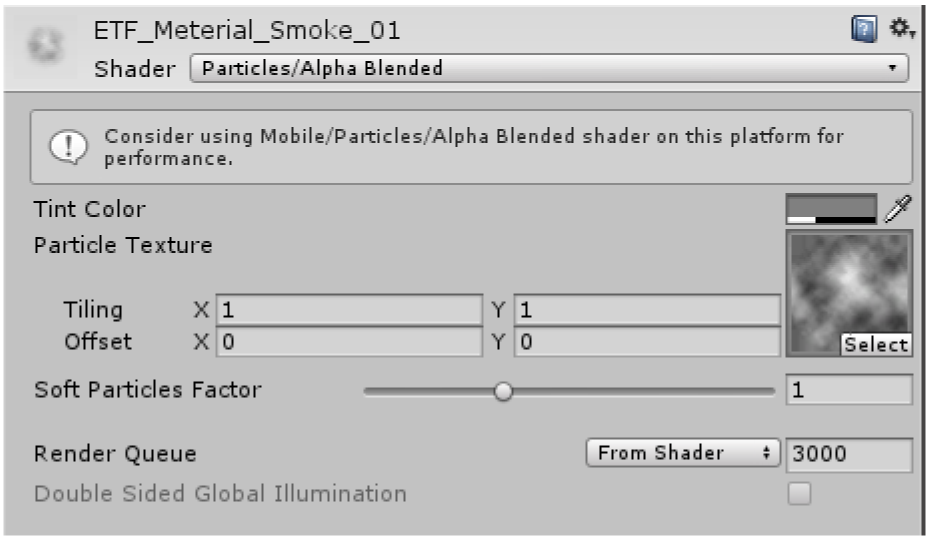
- ➢ The flame at the center of the explosion

This particle system manages its material as a 2*2 sheets and uses one of them to generate particles from a single point. It has a velocity roughly upwards to make the uprising. During its lifetime it will decelerate, rotate, slowly increase its size and change its color from light yellow to black and finally disappear.


ETF_Meterial_Explode_01
Shader Particles/Additive
Consider using Mobile/Particles/Additive shader on this platform for performance.
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue From Shader 3000
Double Sided Global Illumination

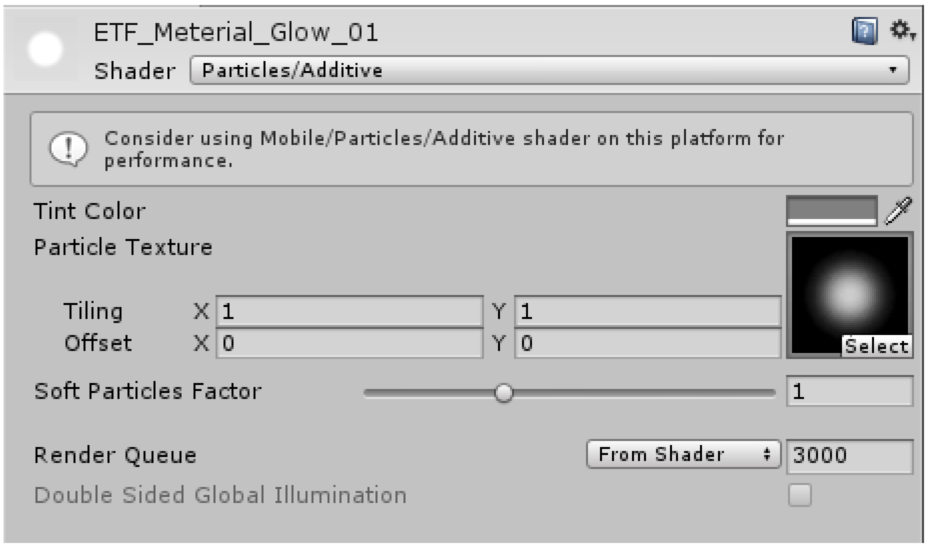
➢ The black smoke

This particle system is quite similar to the flame. Though it generates particles randomly distributed on a hemisphere surface without deceleration during lifetime. Its color change is from invisible to black and finally to invisible again.


ETF_Meterial_Smoke_01
Shader Particles/Alpha Blended
Consider using Mobile/Particles/Alpha Blended shader on this platform for performance.
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue From Shader 3000
Double Sided Global Illumination

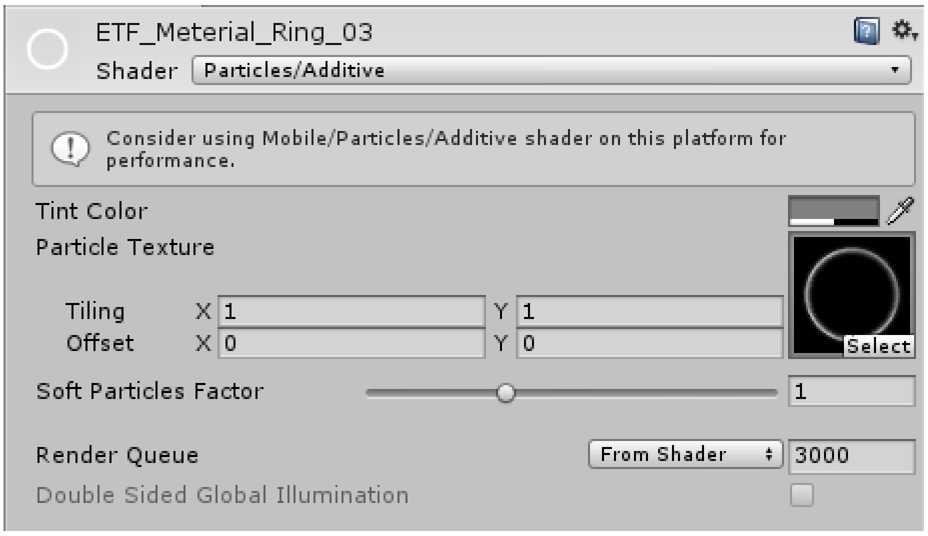
➢ The glow ball at the center of the explosion

This particle system is simply one particle that is generated at the center, increases its size and changes its color from white to invisible.


ETF_Meterial_Glow_01
Shader: Particles/Additive
Consider using Mobile/Particles/Additive shader on this platform for performance.
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue: From Shader 3000
Double Sided Global Illumination

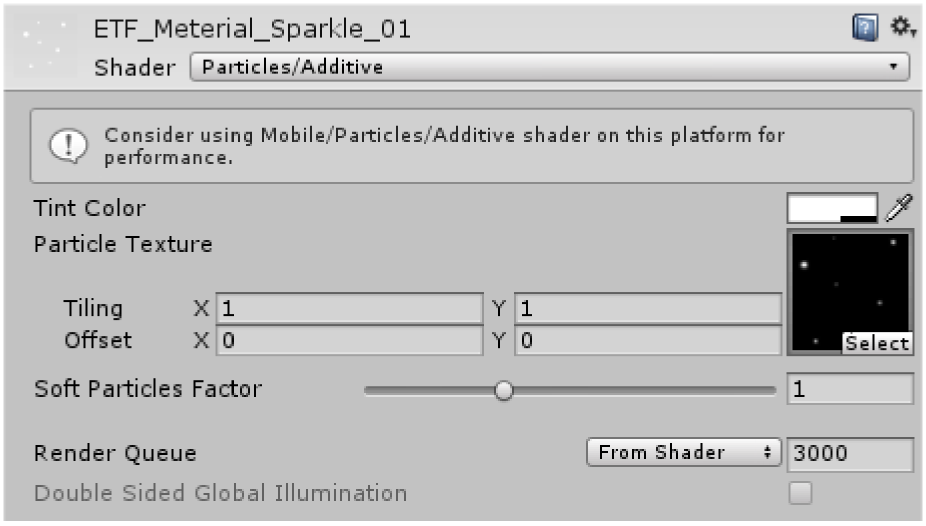
➢ The rings presenting the blast waves

This particle system generates particles along a line segment towards up, having a velocity upwards. The particles will increase the size and gradually disappear over lifetime.


ETF_Meterial_Ring_03
Shader: Particles/Additive
Consider using Mobile/Particles/Additive shader on this platform for performance.
Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue: From Shader 3000
Double Sided Global Illumination

➢ The splashing sparkles

This particle system generates particles distributed on a cone surface with an initial speed upwards and outwards. They will finally fall down because they are affected by a small gravity factor. They will get smaller and the color will gradually become invisible over lifetime.

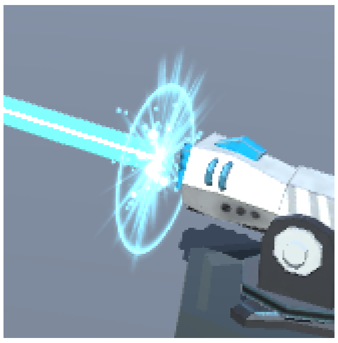


ETF_Meterial_Sparkle_01
Shader: Particles/Additive

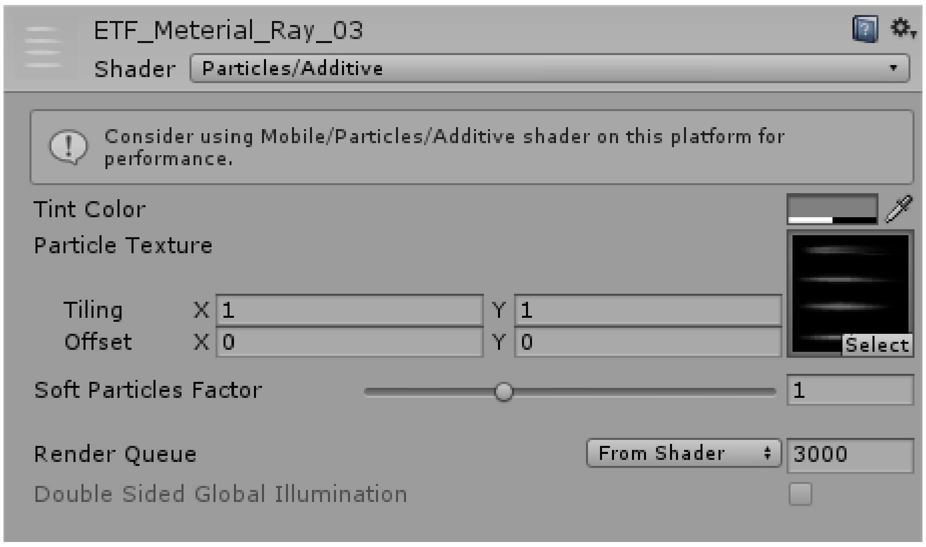
Consider using Mobile/Particles/Additive shader on this platform for performance.

Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue From Shader 3000
Double Sided Global Illumination

5.3.3. The burst during laser emission



This effect consists of 2 particle systems. One is the sparkles just like other sparkles, another is the scattered rays, which generates a lot of particles of ray distributed uniformly on a circle. The particles disappear by changing the color to invisible in very short time.


ETF_Meterial_Ray_03
Shader Particles/Additive

Consider using Mobile/Particles/Additive shader on this platform for performance.

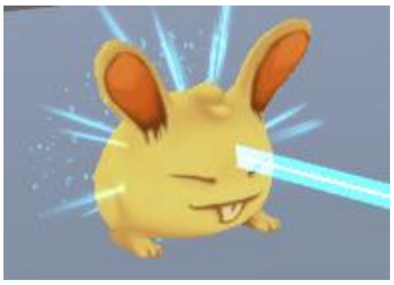
Tint Color
Particle Texture

Tiling X 1 Y 1
Offset X 0 Y 0

Soft Particles Factor 1

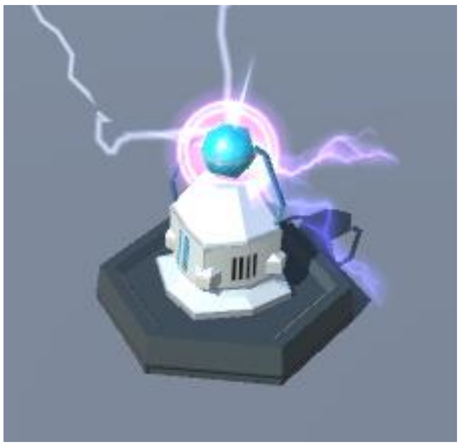
Render Queue From Shader 3000
Double Sided Global Illumination

5.3.4. The burst of laser hit



This effect is almost the same as the burst of laser emission but with more sparkles, larger rays that scattered to all directions.

5.3.5. Pole electrical emission

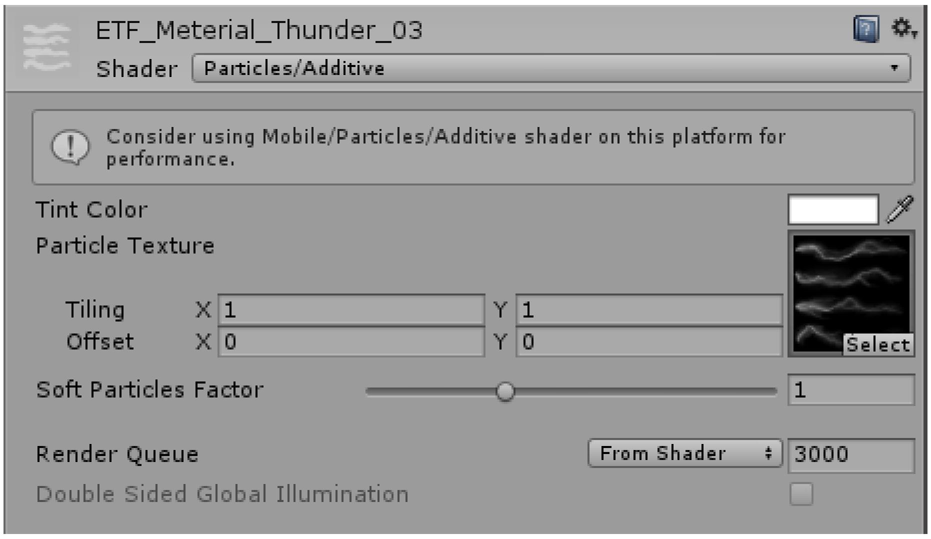


This effect consists of 4 particle systems.

➢ The glow at the center. This is similar to the one in explosion.

➢ The lightnings

This particle system generates particles distributed on a sphere surface. They increase the size slowly and change the color to invisible over lifetime.



ETF_Meterial_Thunder_03
Shader: Particles/Additive

Consider using Mobile/Particles/Additive shader on this platform for performance.

Tint Color
Particle Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Soft Particles Factor 1
Render Queue From Shader 3000
Double Sided Global Illumination

➢ The “sphere”

This particle system actually uses the material of the ring, which presents the blast waves. What differs is that it always faces to the camera so it looks like a sphere.

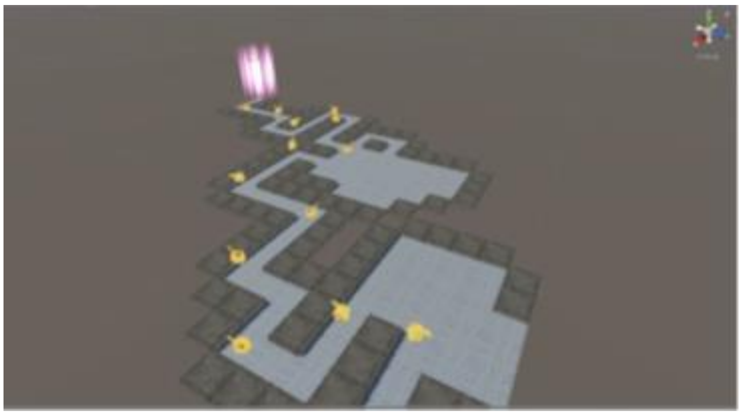
➢ The rays.

This is similar to the one in the burst of laser hit.

5.4. Enemies

5.4.1. Enemy Generator

The enemy generator generates different types of enemy waves. We also make the generator more completed in having ability to make a single wave carrying different enemies of certain numbers to improve the difficulty of the defense.



Different types of enemies

The models and the animations are downloaded from Assets Store of Unity for free.

i. Walking enemies: Walks along the route.

- Rabbit

- Ghost

ii. Flying enemies: Fly directly to the terminal.

- Bat

5.4.2. Real-time enemy navigation

To improve the playability and get the effect of advancing wit, we have implemented the function that the tower can be placed on the path to block the route, so that we allow player to lead the enemies to take long detours. But The built-in navigation system cannot tolerate dynamic obstacle changes in the map. Therefore, we implement our own algorithm for the navigation system.

The most commonly used algorithm for navigation is A*, but it is too slow in our game since we require multiple enemies to react to the variation in a possibly large map very quickly. So we implement the D* Lite algorithm, which is faster and more suitable for dynamic environment.

5.5. Interaction

5.5.1. Tango implementation

- Map Terrain Generation

In this game, we need use Tango to find user-defined plane at touch position once clicking on smartphone screen, so we can generate our map on this plane. Through tango, we can a get point cloud where each point contains information of real world location. And we can use this point could to find desired plane in two passes. First, all points in tango point cloud within certain pixel distance to UV coordinates after projection are kept. Then a plane is fit to the subset cloud using RANSAC. After the initial fit all inliers from the original cloud are used to refine the plane model.

- Tower Building Position

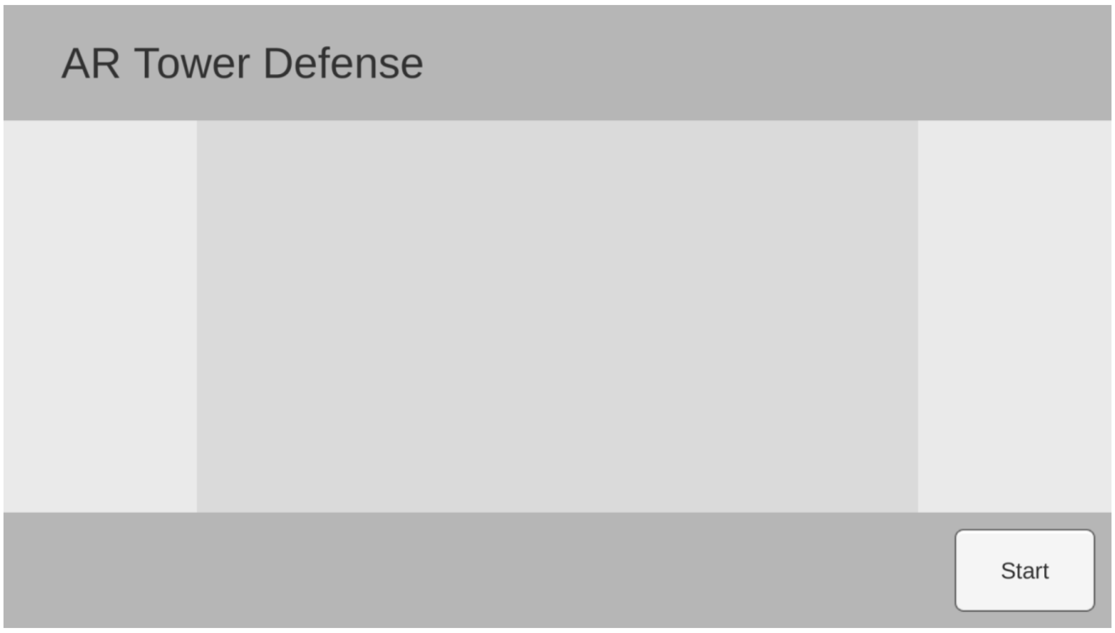
After generating the map on a surface, the player can move around to observe the map and decide where to build the towers before starting the first enemy wave. It uses the motion tracking to calculate the new relative position of the device to the virtual objects in every frame to render the view of the virtual objects. And for every several frames it will recognize the location using area learning to adjust the moving trajectory. Using this method, we can keep the map at that surface while the user holds the device and walks around.

We use raycast from the center of the screen to hit the tile to select.Afterwards the player can click the button "Build Tower" to build above the selected tile.

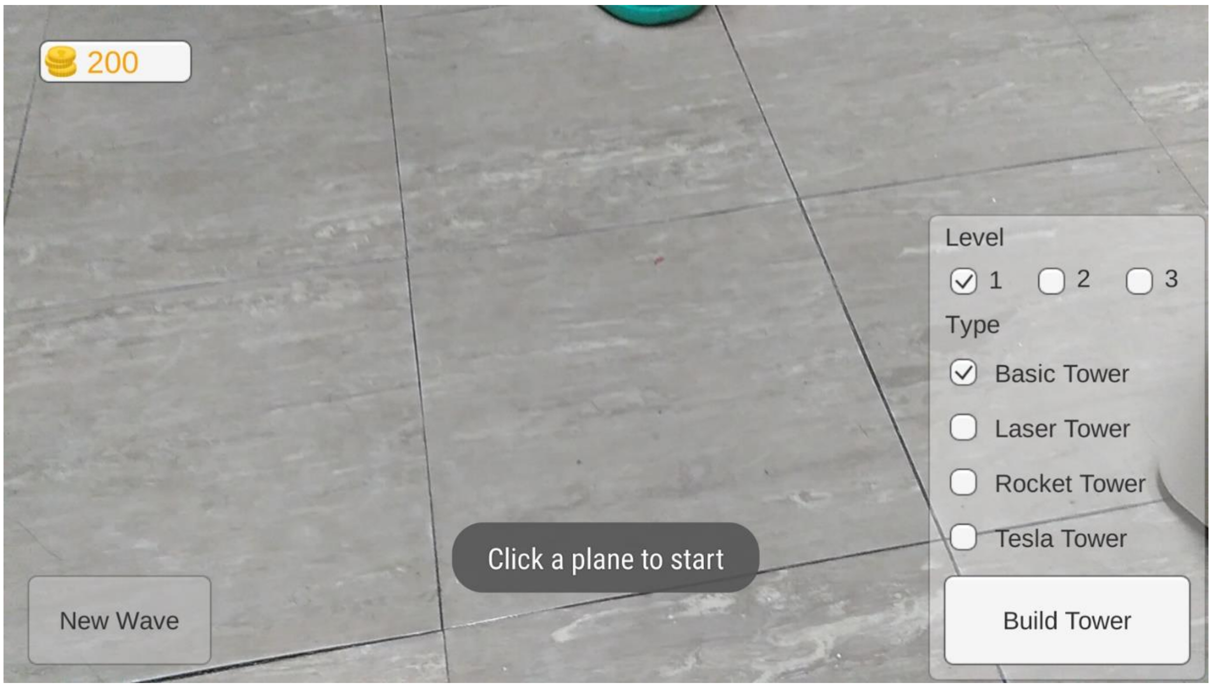
5.5.2. UI

Currently we use the basic UI components provided by Unity to compose our Application UI, which can be improved to more beautiful and informative with our own pictures in our further plan.

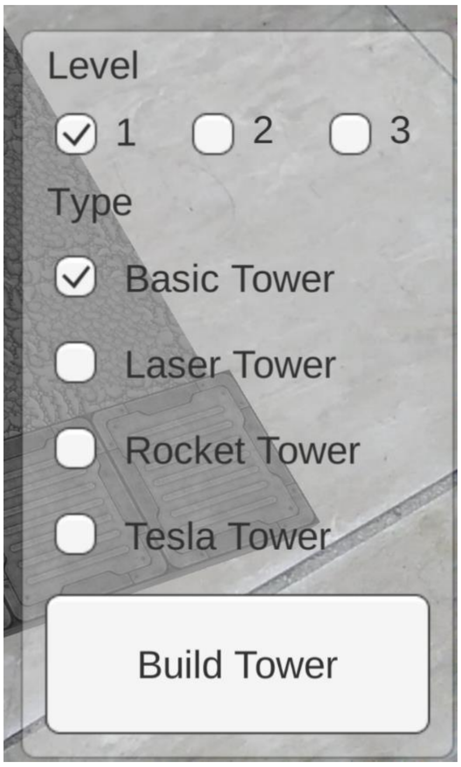
- The Initial Interface



- The Play Interface

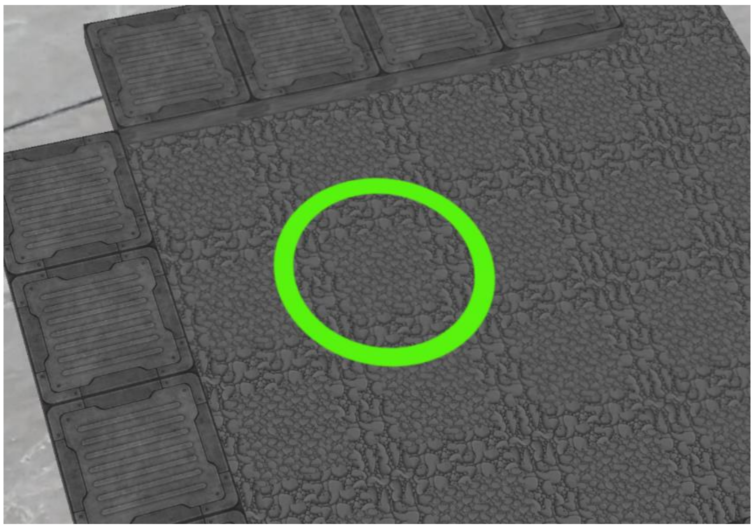


The building settings

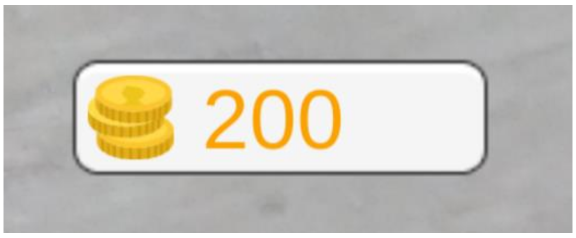


Player can select different types and levels of the tower and essentially click “Build Tower” to build above the selected tile

- The circle to highlight the selected tile



- The coin counter



- The Button to start new wave

Player can click this button to let the enemy generator start generating enemies. This button will be enabled after the map is generated and will be disabled until all the current wave’s enemies have been killed.

6. Conclusion

During this term's project development, we have obtained a lot of knowledge about Tango and Game Development. And we find AR has great potential in future Computer Human Interaction and can be applied to fascinating instant communication.

Through the research and implementation of Tango, we find that it might show no great advantages in mobile entertainment. Compared to traditional mobile games, AR games has higher difficulty in operation. The players need to hold the phone carefully, move their arms to turn the camera's direction to observe the surroundings, and even walk around to play the game, which consumes much more energy, so it will not last for the players to play. And the novelty will also fade away quickly.

Mobile AR also has less interactivity compared with Head Mounted Virtual Reality devices such as HTC Vive with hand shanks as controller while mobile AR only handles finger movements on the screen like touches and wipes.

This term we have only only implement the AR in finding planes on the surface of physical objects. We didn't immersively combine the environment to the game itself. For example, the map is just generated on the surface that the user clicks; it will not keep parallel with the edges of the surface; the map will usually go beyond the range of the surface.

We plan to explore more in next term about AR. A great AR application should be combined with the reality, make responses when the interaction between the user and the physical objects, and learn the environmental changes.

7. Future Plan

Area Reconstruction for Map Generation

To exploit the Augment Reality, we plan to combine the Area Reconstruction function to map generation.

We have evaluated the difficulty and concluded 2 ways to approach it:

- Uses the point cloud to generate a invisible mesh of the real environment and the enemies just move above the “real” desktop, that is, no virtual terrain generation.

- Prepare the point cloud in several directions to construct the whole 3D model of the environment. And then maybe make the protrusions to become the obstacles and the flat surface to become paths and rooms. And the tiles can also be placed parallel or perpendicular to the edges of the surface and the walls. We also need to calculate the size of the surface and decide how large the map size should be and where to place the start position as well as the end position.

Multi-player Mode Sharing the Environmental Information through Internet

As we are going to explore the new interaction methods through the utility of

Augment Reality, we want to add communication, cooperation and competition between players.

After we get the second device supporting Tango, we will start building our game server to control the data communication and storage among devices.
We considered the following 2 scenarios:

- Different players are physically in the different room, and one of them shares the virtual objects to the others.

- Different players are physically in the same room to watch the same virtual objects, so they should also share the environmental information (area description).

For the first scenario, it is easier because it does not matter where and what direction the virtual objects are placed in the different room.

The difficulty for the second scenario is to make sure the players see the same virtual objects placed in the same direction, so the area recognition is rather important for the shared environmental information. We need to extract the features of this area and mathematically compare the data from different devices to coincide the area.

Better UI

We will contribute some efforts to draw the patterns of UI. Also we need to design how to display more necessary information and not occupy too much area on the screen to make convenience of the player.

8. Citation

[1] Azuma, Ronald T. "A survey of augmented reality." *Presence: Teleoperators and virtual environments* 6.4 (1997): 355-385.

[2] Schmalstieg, Dieter, and Tobias Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[3] Google. " Tango Concepts." 2017. *Tango Developer Documentation*. Visited at 28 11 2017: https://developers.google.com/tango/developer-overview

[4] Google. " Motion Tracking." 2017. *Tango Developer Documentation*. Visited at 28 11 2017: https://developers.google.com/tango/overview/motion-tracking

[5] Google. " Area Learning." 2017. *Tango Developer Documentation*. Visited at 28 11 2017: https://developers.google.com/tango/overview/area-learning

[6] Google. " Depth Perception." 2017. *Tango Developer Documentation*. Visited at 28 11 2017: https://developers.google.com/tango/overview/depth-perception

9. Reference

Chaosemer. “Project Tango UnitySDK Examples” GitHub. GitHub, 2013. Web. 15 Jan 2013
<http://gist.github.com/admsyn/3452792>

Unity. Particle Systems: Unity Technologies. Unity, 2017. Web. 12 Jul 2017.
<https://docs.unity3d.com/560/Documentation/Manual/ParticleSystems.html>

Unity. Line Renderer: Unity Technologies. Unity, 2017. Web. 12 Jul 2017.
<https://docs.unity3d.com/560/Documentation/Manual/class-LineRenderer.html>

Unity. Animator Controller: Unity Technologies. Unity, 2017. Web. 12 Jul 2017.
<https://docs.unity3d.com/560/Documentation/Manual/class-AnimatorController.html>

Unity. Using Animation Events: Unity Technologies. Unity, 2017. Web. 12 Jul 2017.
<https://docs.unity3d.com/560/Documentation/Manual/animeditor-AnimationEvents.html>

Unity. Getting started with Android development: Unity Technologies. Unity, 2017. Web. 29 Nov 2017.
<https://docs.unity3d.com/560/Documentation/Manual/android-GettingStarted.html>

Microsoft. Getting Started with Visual Studio Tools for Unity: ghogen, Saisang, etc.Microsoft. Web. 4 July 2017.
<https://docs.microsoft.com/en-us/visualstudio/cross-platform/getting-started-with-visual-studio-tools-for-unity>

Google. Getting Started with Unity for Tango: Google. Google Inc, 2017. Web. 21 Jun 2017.
<https://developers.google.com/tango/apis/unity/>