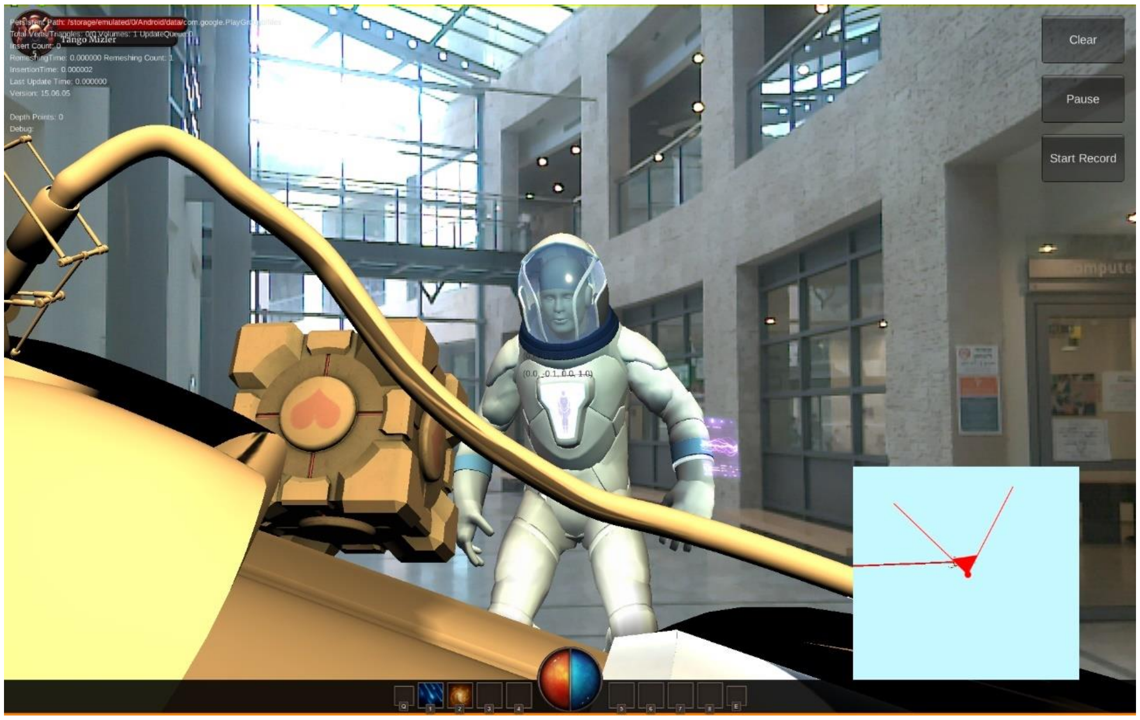


```
Persistent Path: /storage/emulated/0/Android/data/com.google.PlayGr[illegible]
Total Verts/Triangles: 0/0 Volumes: 1 UpdateQueue: 0
Insert Count: 0
RemeshingTime: 0.000000 Remeshing Count: 1
InsertionTime: 0.000002
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 0
Debug:
```

MultiAR Project

Michael Pekel, Ofir Elmakias

[GIP] | [234329]

Supervisors

Dr. Matan Sela

Mr. Yaron Honen

Assistants

Alexander Porotskiy

Summary

MultiAR is a multiplayer quest (Outdoor Real Time Multiplayer AR Shooting) Game targeting the augmented reality platform, Google Tango. This game simulates the game mechanics in PokémonGO coop with a random opponent, but with fantasy elements and interesting game mechanics. The key experiences from this game are the game mechanics, power-ups, social element and challenging tasks. The game is made more immersive by using the capabilities of VR and AR platforms by scanning the play field in RT.

Introduction

We created a mobile game that runs on the Google Tango, developed in Unity 5 - graphic engine, scripted with C# in Visual Studio 2015 and MonoDevelop environment.
Unity is a cross-platform game engine that is used to develop video games for PC, consoles, mobile devices and websites.

Tango is a platform that uses computer vision to give devices the ability to understand their position relative to the world around them. It's similar to how you use your eyes to find your way to a room, and then to know where you are in the room and where the floor, the walls, and objects around you are. These physical relationships are an essential part of how we move through our daily lives. Tango gives mobile devices this kind of understanding by using three core technologies: Motion Tracking, Area Learning, and Depth Perception.

Motion Tracking means that a Tango device can track its own movement and orientation through 3D space. Walk around with a device and move it forward, backward, up, or down, or tilt it in any direction, and it can tell you where it is and which way it's facing.

With Area Learning turned on, the device not only remembers what it sees, it can also save and recall that information. When you enter a previously saved area, the device uses a process called localization to recognize where you are in the area. This feature opens up a wide range of creative applications. The device also uses Area Learning to improve the accuracy of Motion Tracking.

Tango gives your mobile device the same ability. With Motion Tracking alone, the device "sees" the visual features of the area it is moving through but doesn't "remember" them.

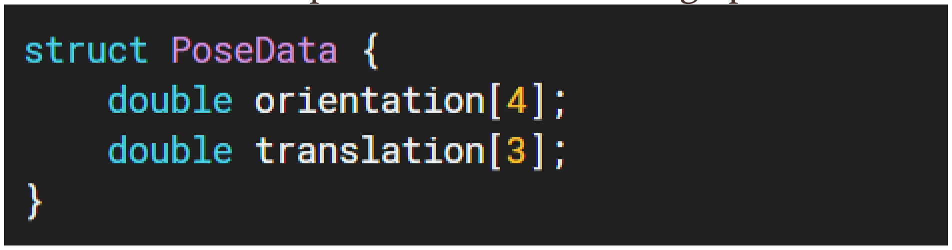
With depth perception, your device can understand the shape of your surroundings. This lets you create "augmented reality," where virtual objects not only appear to be a part of your actual environment, they can also interact with that environment.

What Are Tango Poses?

As your device moves through 3D space, it calculates where it is (position) and how it's rotated (orientation) up to 100 times per second. A single instance of this combined calculation is called the device's pose. The pose is an essential concept when working with motion tracking, area learning, or depth perception.

To calculate the poses, you must choose base and target frames of reference, which may use different coordinate systems. You can view a pose as the translation and rotation required to transform vertices from the target frame to the base frame.

Here is a simplified version of a Tango pose struct in C:



```c
struct PoseData {
    double orientation[4];
    double translation[3];
}
```

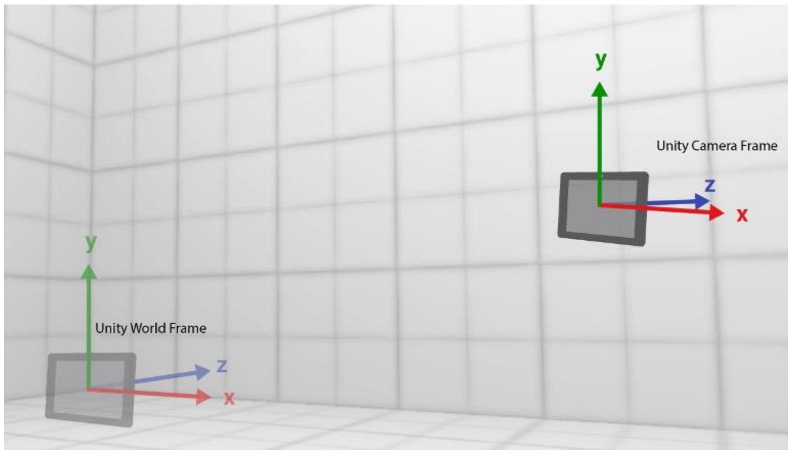
The two key components of a pose are:

- A quaternion that defines the rotation of the target frame with respect to the base frame.
- A 3D vector that defines the translation of the target frame with respect to the base frame.

Unity Coordinate System

The Tango API returns pose data as a rotation and translation between two coordinate frames with X, Y, and Z values.

The Unity World Frame and Camera Frame both use a left-handed coordinate system: the X axis is horizontal with the positive direction to the right, the Y axis is vertical with the positive direction pointing up, and the Z axis is depth (forward and back), with the positive direction pointing away from the user. In this system, the camera looks in the direction the Z axis is pointing.



Converting Tango Pose data into the Unity Coordinate System

We can handle Unity Coordinate System conversion as follows. The first transformation converts between the Project Tango START_OF_SERVICE (SS) coordinate frame and the Unity World (UW) coordinate frame.

$$ {}^{UW}_{SS}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The second transformation converts between the Unity Camera (UC) coordinate frame and the Tango Device (D) coordinate frame. You'll notice that this is not the identity matrix because Unity uses a left-handed coordinate system.

$$ {}^{D}_{UC}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Multiply the first conversion and then multiply the second conversion to get a matrix that converts pose data between the Unity Camera (UC) coordinate frame and the Unity World (UW) coordinate frame.

$$ {}^{UW}_{UC}T = \left({}^{UW}_{SS}T * {}^{SS}_{D}T \right) * {}^{D}_{UC}T $$

Characters

Since the game is to simulate the real world experience, the characters in this game retain their identities as in the real world. We added portal gun which the user can control, carry, shoot & open portals with.

Components

AR Environment and tracking using Google Tango plugin
Multiplayer networking using own implemented Client-Server + API
Google VR DayDream Plugin integrate controller emulators to interact with objects in Mixed-Reality
Physics using the Unity’s physics engine

<u>Headset</u>
The Tango Development kit itself uses us as a Gateway between Real world & Mixed Reality, where it uses the camera to project what goes in front of the player on the screen & augments virtual objects in the scene.

<u>Controllers</u>
Mobile devices run emulator app to function as a Gyro tracked handheld controllers to vividly manipulate objects, interact with precision, communicate and experience immersive environments

Virtual Reality Setup

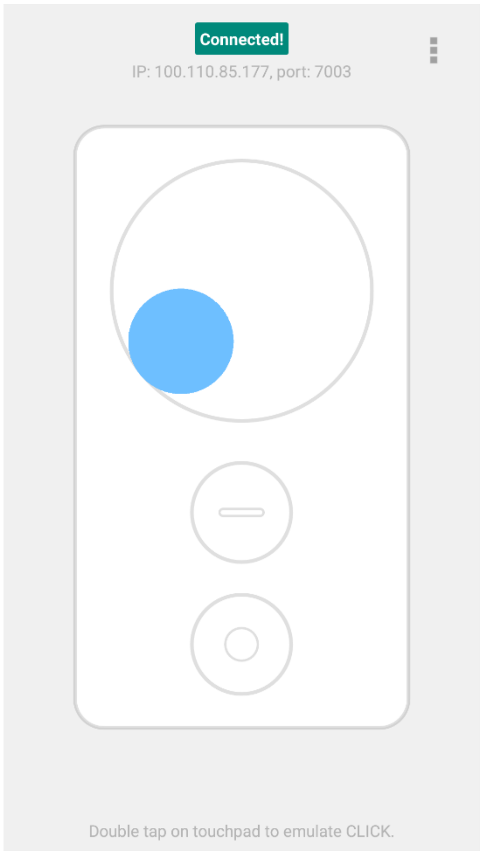
Tango SDK
https://developers.google.com/tango/downloads

The Controller Emulator
https://developers.google.com/vr/daydream/controller-emulator

Google VR SDK for Unity
https://developers.google.com/vr/unity/

Unity3D Game Engine
https://store.unity.com/download?ref=personal

Application User Guide


Connected!
IP: 100.110.85.177, port: 7003
Double tap on touchpad to emulate CLICK.

Interface

The Controller Emulator UI contains these elements:

- Connection indicator (top): This text indicates whether the emulator is connected to a headset phone. It also shows network information for the controller phone.
- Touchpad area (large circle): This area emulates the touchpad area of a controller. It does not support multi-touch.
- Touchpad click emulation: The touchpad can be clicked by pressing down on it. This is considered to be a button. Clicking is emulated by double-tapping the touchpad in the Controller Emulator and will send Click button events to the app.
- App button (immediately below touchpad): This button sends App button events to the app.
- Home button (bottom): This button is reserved for system use and cannot be used by app. It is also used to recenter the controller.

Gestures

The touchpad area functions as 5 different UI's to make it more immersive sense of control. We implemented Swipe detector to distinguish between actions same as compass rose, therefore we can detect 4 different swipes: up, down, left & right. (the fifth UI is the Touchpad click)

➔ Swipe UP – throws the shells, where the force is controlled by the swipe speed
➔ Swipe Down – clear the portals
➔ Swipe Left – Open left portal
➔ Swipe Right – Open right portal
➔ App button – functions as general purpose button (Accept Quest msgs)

Notice: After room scanned & calibration, if someone is moving inside the room it builds a mesh around him and stays as a physical object in the scene.

Features and Implementations (UI/UX)

Basically, we added mesh builder and occlusion both together to get immersive effects. The mesh renderer is entirely contained in a mesh rendering script. The mesh is represented as a multitude of in-engine Game Objects with shader and transform components. The shader gives the mesh the color of the object being represented. The transform determines the position and orientation of the mesh in the application world. The transform is extremely important as an accurate transform determines the accuracy of the model generated.

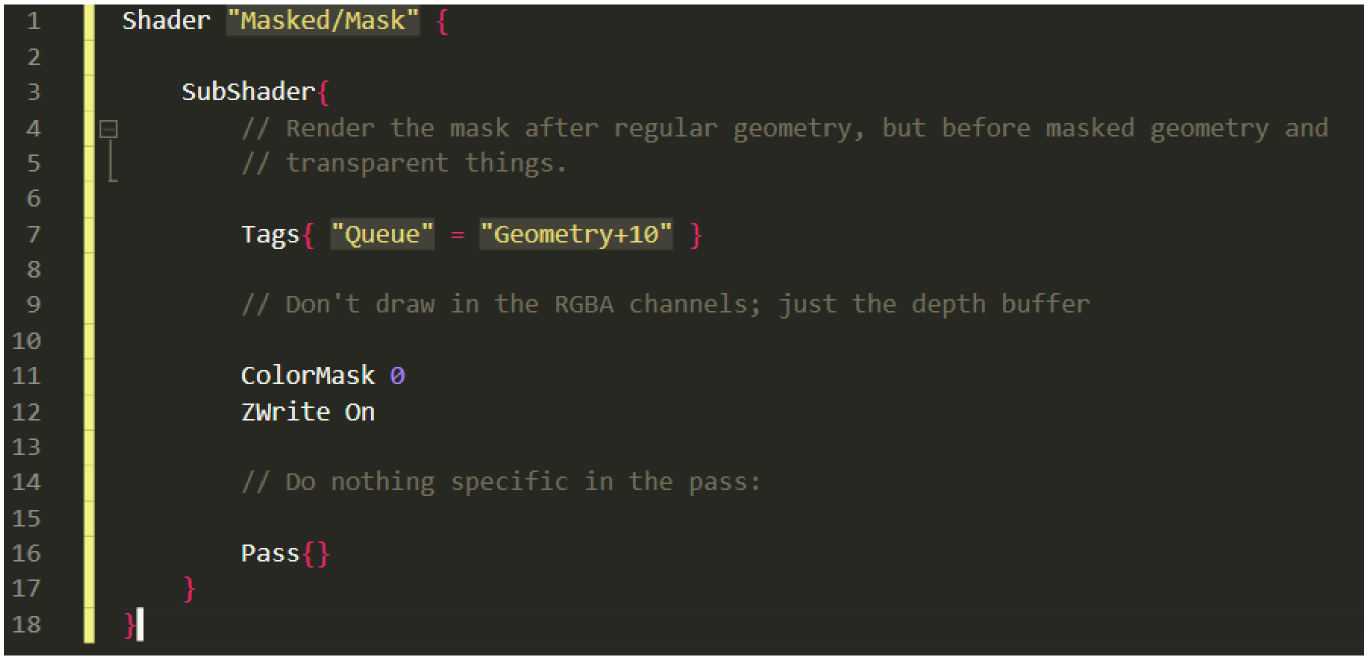
The mesh rendering processing goes through quite a few steps. The mesh renderer subscribes to the Tango's depth sensor callback. The Tango feeds the mesh renderer depth data allowing the renderer to find flat surfaces. Each surface is then transformed into a grid in which a square on that grid can then be meshed. Depending on the resolution determined beforehand (we found 5 cm to be efficient) the grid will have differently sized squares. A point cloud is generated to coloring each mesh to better represent what is being mapped.

To improve the user experience, the computationally heavy mesh rendering is done on a separate thread. When indices are seen that require meshing, they are pushed into a queue data structure to await their turn. This backlog is processed for a small portion of the computation cycle every cycle. Of course, this means that mapping a brand-new area creates a surge in meshes that need to be processed.

By keeping the Tango focused on a previously rendered area or by pausing the rendering process through the GUI, the backlog can be processed much more quickly. The size of the mesh can be controlled, if you want to get a more detailed result, increase voxel resolution in MeshManager and MeshingCube.

Then we make the mesh be translucent. there are lots of ways to do that. What we did is to use mask shader, so the mesh cube's material is mask, and the game object (bullet) has SetRenderQueue on them. more details here:

http://wiki.unity3d.com/index.php?title=DepthMask



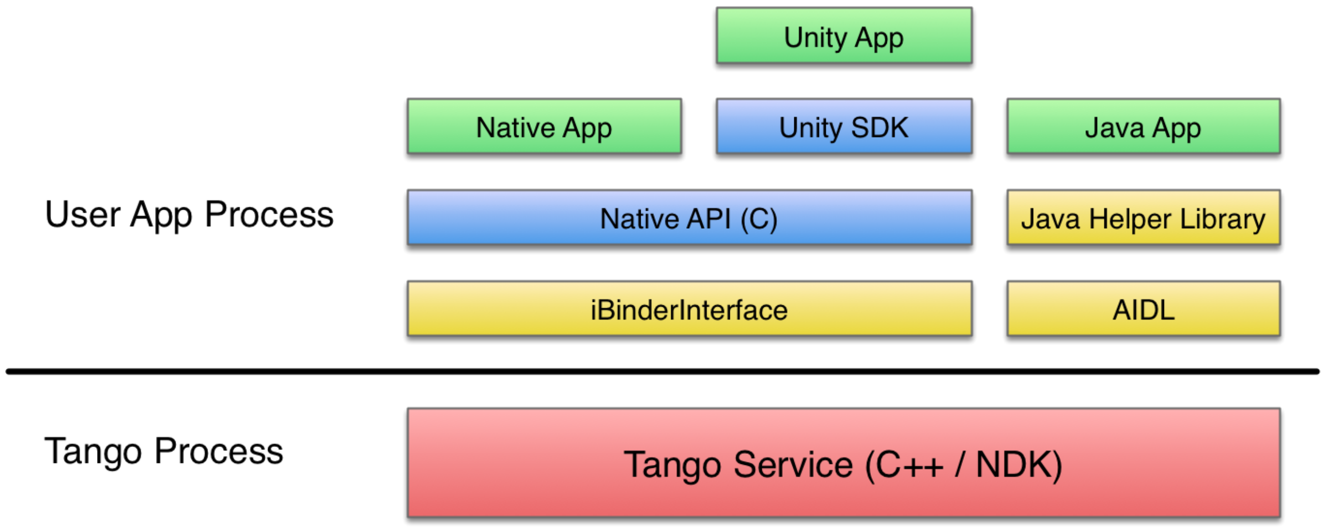
```
Shader "Masked/Mask" {

    SubShader{
        // Render the mask after regular geometry, but before masked geometry and
        // transparent things.

        Tags{ "Queue" = "Geometry+10" }

        // Don't draw in the RGBA channels; just the depth buffer

        ColorMask 0
        ZWrite On

        // Do nothing specific in the pass:

        Pass{}
    }
}
```

Server

We decided to use the centralized server. Basically, the "host" device is the server and if we add an object, we would add that to the state of the server. Periodically all the devices are polling for updates of state. How quickly you want to poll depends on the use case. At certain events, we made it so that it could send updates to the server. The main info we upload is the vector and quaternion for position and orientation and a string with the name of the object we want to place and bullets when player shoot is recorded. The only thing we need to keep in sync with devices is just those position and orientation information is only 7 numbers, which is not as expensive. The only time this becomes a problem is if there are lots of objects, because then it becomes a lot of information that needs to be polled very quickly.

API Overview

This is the current Tango application development stack:



TangoService is an Android service running on a standalone process. It uses standard Android Interprocess Communication to support apps written in Java, Unity, and C. TangoService performs all of the main Tango technologies, such as motion tracking, area learning, and depth perception. Applications can connect to TangoService through APIs. We chose to work with Unity SDK which is great for making games and other programs requiring 3D visualization.

Mechanics

Each user holds tight the Tango device while it initialized and pointed to the same orientation and place. Our mission is to defeat the bad guys (3D creatures) in the scene and cooperate while moving through the quests (each quest is followed by unique message). The players move in the real world holding the Tango Device and the controller to navigate and shoot the enemies. They can carry objects and take it with them while they moving while it's necessary to accomplish the quest.

The final quest is to build in the right order the puzzle in order to win the game.

- **Teleporter**

  Provides to the user an ability to traverse portable objects, freely in the scene, of various heights as the endpoint can be curved on top of objects that are not necessarily visible to the user. The user can open portals (Orange & Blue) and throw portable objects from one side to another. The teleporter consists of Red straight Laser Pointer that emits a straight line, made out of game objects from the end of the controller to a raycast hit surface (at any height or place).

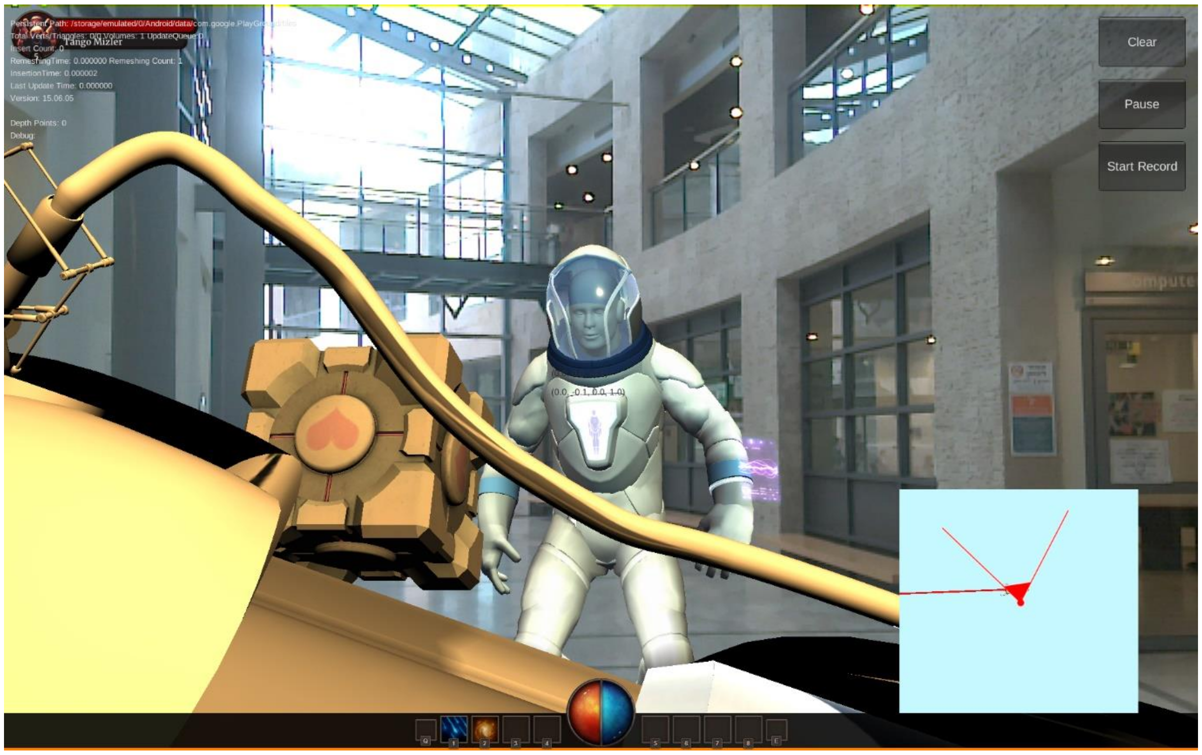
❖ **Ray cast**

Finds closest object by shooting a ray into the scene and highlights the selected one.

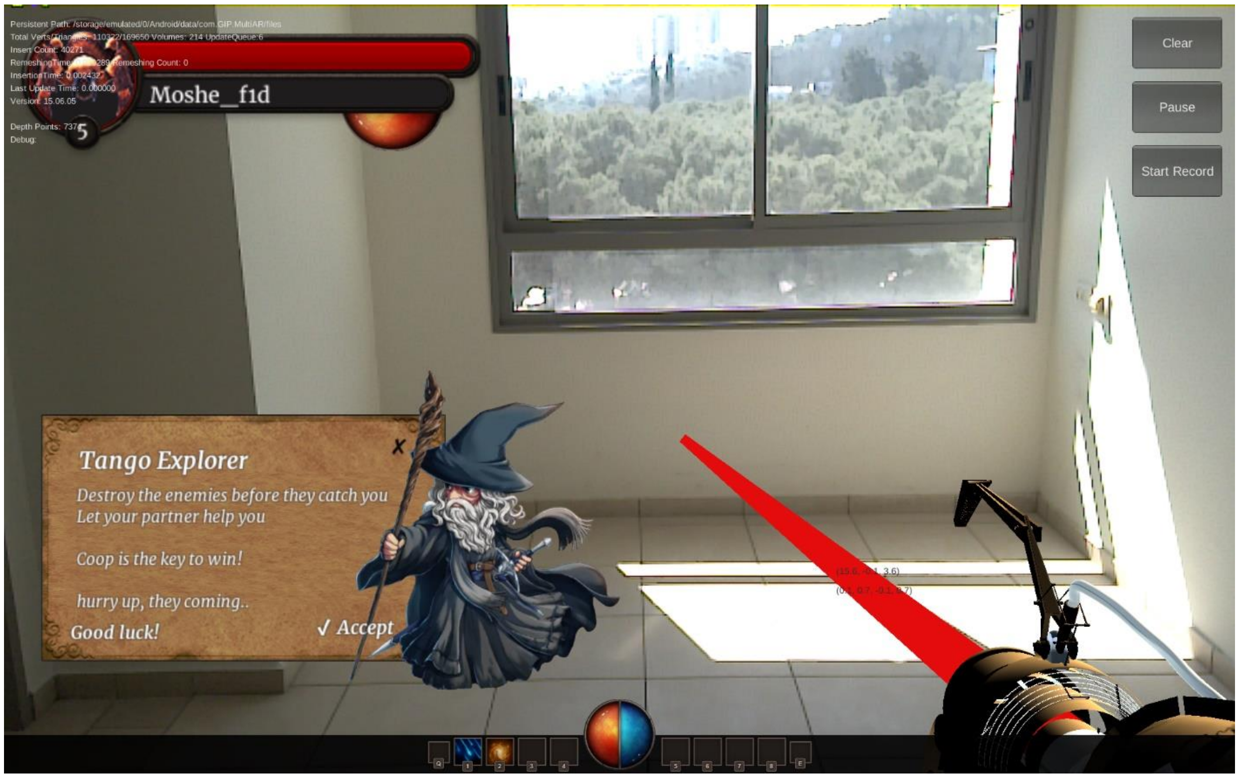
Bibliography

- Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially-Hashed Signed Distance Fields
  http://www.roboticsproceedings.org/rss11/p40.pdf
- Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation
  https://introlab.3it.usherbrooke.ca/mediawiki-introlab/images/b/bc/TRO2013.pdf
- Large-scale, Real-Time Visual-Inertial Localization
  http://people.inf.ethz.ch/sattlert/publications/Lynen2015RSS.pdf
- Accurate, Dense, and Robust Multi-View Stereopsis
  http://www.cse.wustl.edu/~furukawa/papers/pami08a.pdf
- Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images
  http://strands.acin.tuwien.ac.at/publications/2014/hermans-icra14.pdf
- 3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices
  https://www.cvg.ethz.ch/research/3d-modeling-on-the-go/schoeps3dv2015.pdf
- Fast Retina Keypoint
  https://infoscience.epfl.ch/record/175537/files/2069.pdf
- Binary Robust Invariant Scalable Keypoints
  https://www.robots.ox.ac.uk/~vgg/rg/papers/brisk.pdf
- Large-Scale Location Recognition and the Geometric Burstiness Problem
  http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Sattler_Large-Scale_Location_Recognition_CVPR_2016_paper.pdf

Screenshots



```
Persistent Path: /storage/emulated/0/Android/data/com.google.PlayGround/files
Total Verts/Triangles: 0/0 Volumes: 1 UpdateQueue:0
Insert Count: 0
RemeshingTime: 0.000000 Remeshing Count: 1
InsertionTime: 0.000002
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 0
Debug:
```



```
Persistent Path: /storage/emulated/0/Android/data/com.GIP.MultiAR/files
Total Verts/Triangles: 110322/169650 Volumes: 214 UpdateQueue:6
Insert Count: 40271
RemeshingTime: 0.[illegible]289 Remeshing Count: 0
InsertionTime: 0.002432
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 737[illegible]
Debug:
```

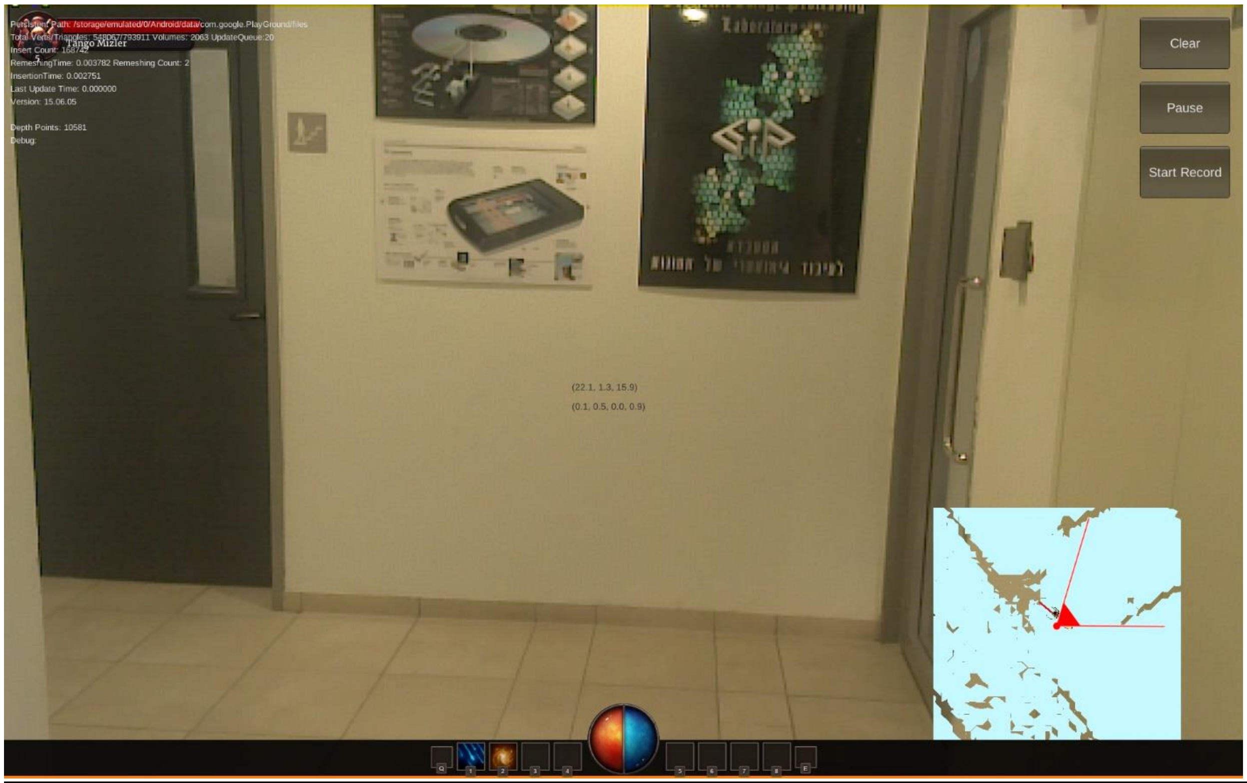
Tango Explorer

Destroy the enemies before they catch you
Let your partner help you

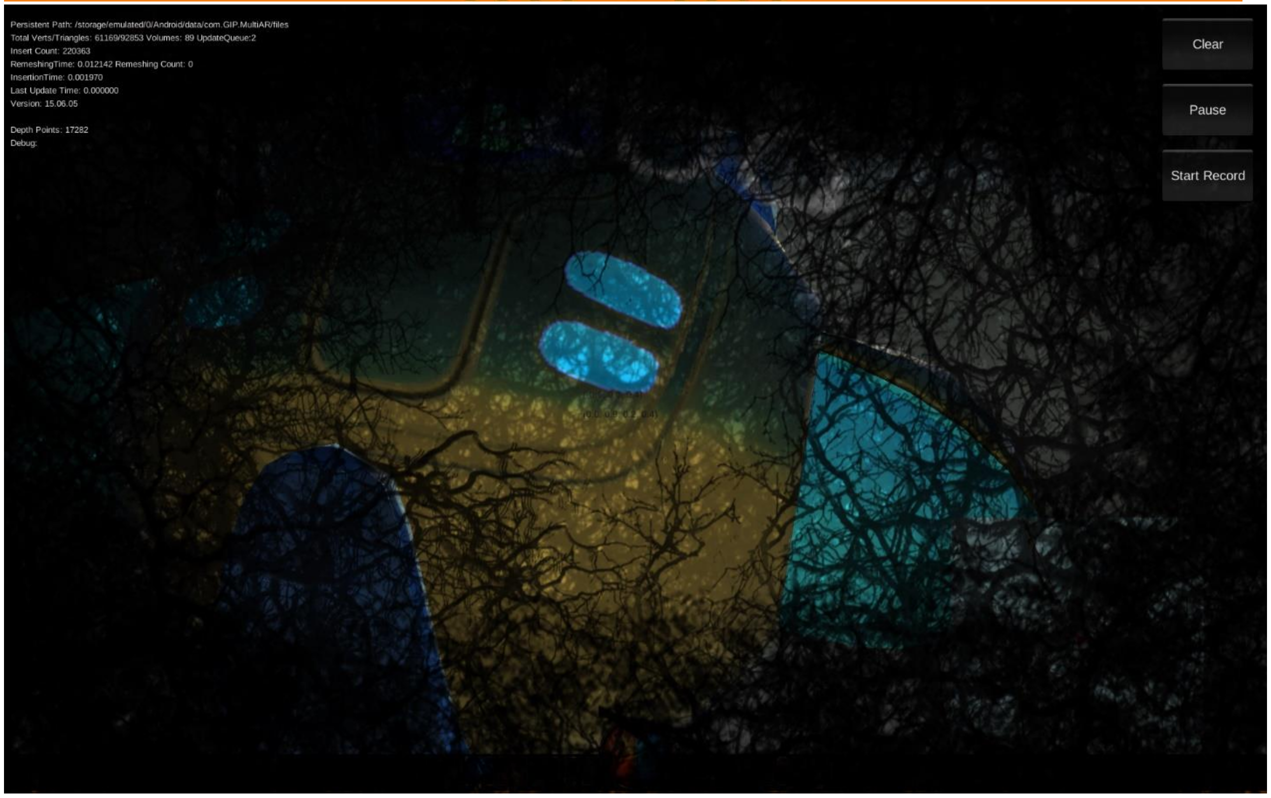
Coop is the key to win!

hurry up, they coming..

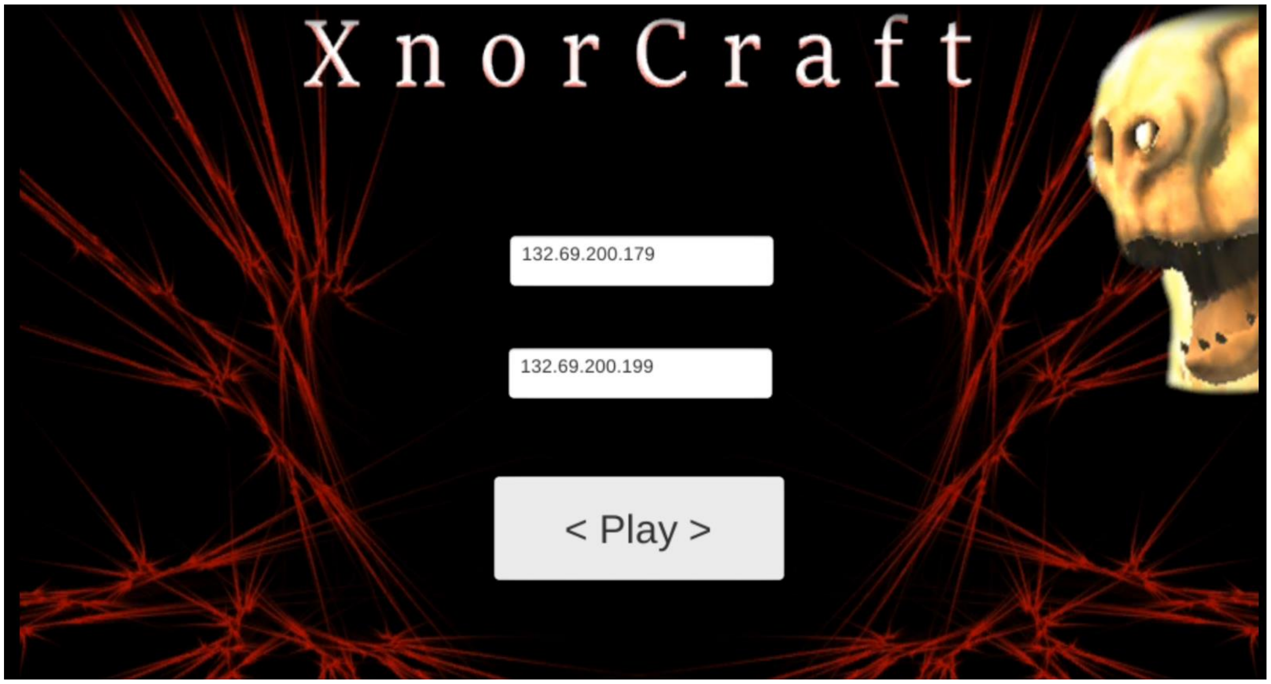
Good luck!

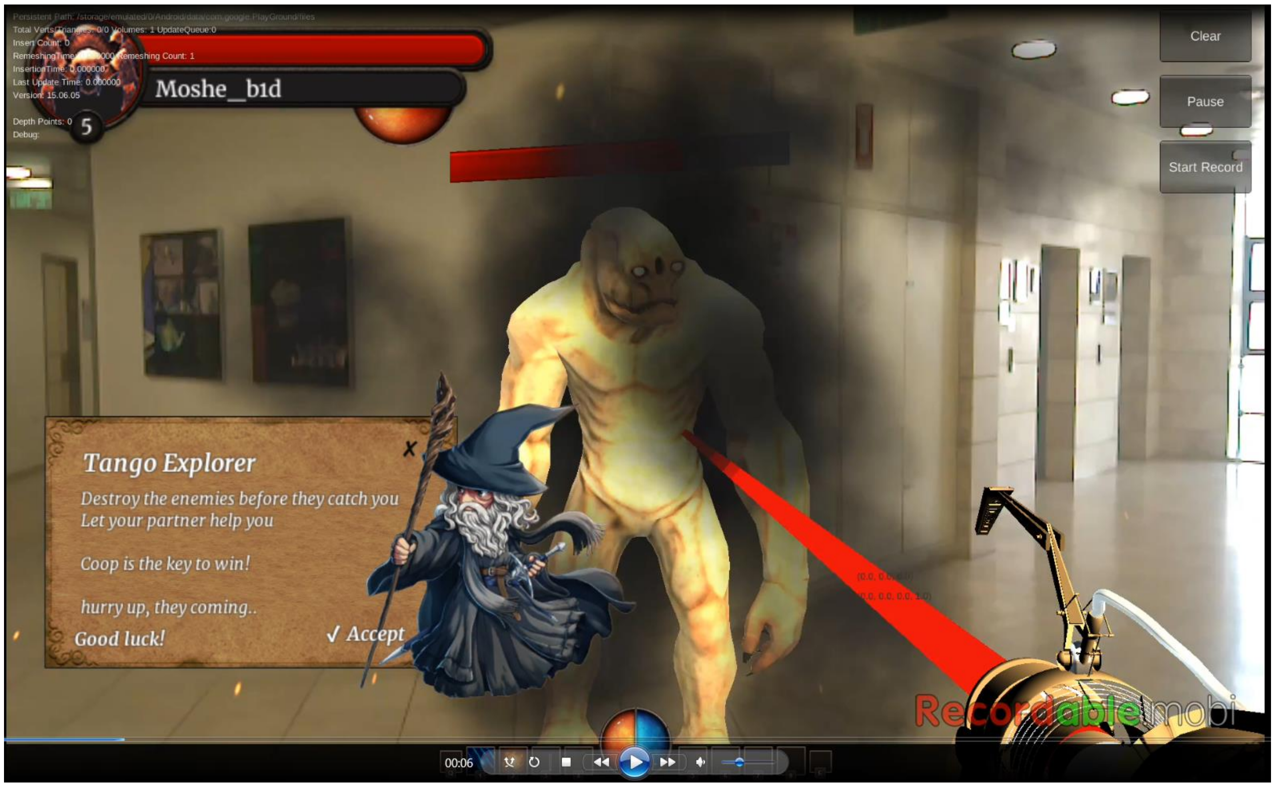


```
Persistent Path: /storage/emulated/0/Android/data/com.google.PlayGround/files
Total Verts/Triangles: 548067/793911 Volumes: 2063 UpdateQueue:20
Insert Count: 168742
RemeshingTime: 0.003782 Remeshing Count: 2
InsertionTime: 0.002751
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 10581
Debug:
```



```
Persistent Path: /storage/emulated/0/Android/data/com.GIP.MultiAR/files
Total Verts/Triangles: 61169/92853 Volumes: 89 UpdateQueue:2
Insert Count: 220363
RemeshingTime: 0.012142 Remeshing Count: 0
InsertionTime: 0.001970
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 17282
Debug:
```




```
Persistent Path: /storage/emulated/0/Android/data/com.google.PlayGround/files
Total Verts/Triangles: 0/0 Volumes: 1 UpdateQueue:0
Insert Count: 0
RemeshingTime: [illegible]000 Remeshing Count: 1
InsertionTime: 0.000000
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 0
Debug:
```

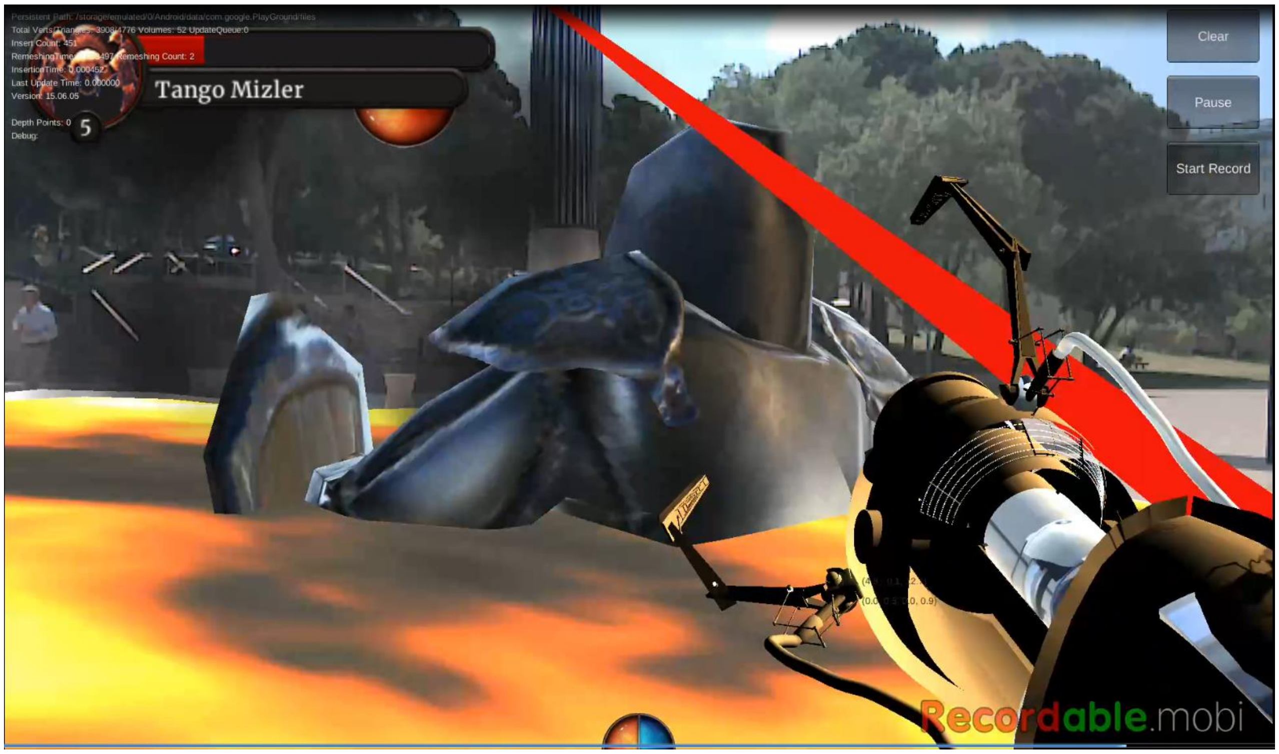
Tango Explorer

Destroy the enemies before they catch you
Let your partner help you

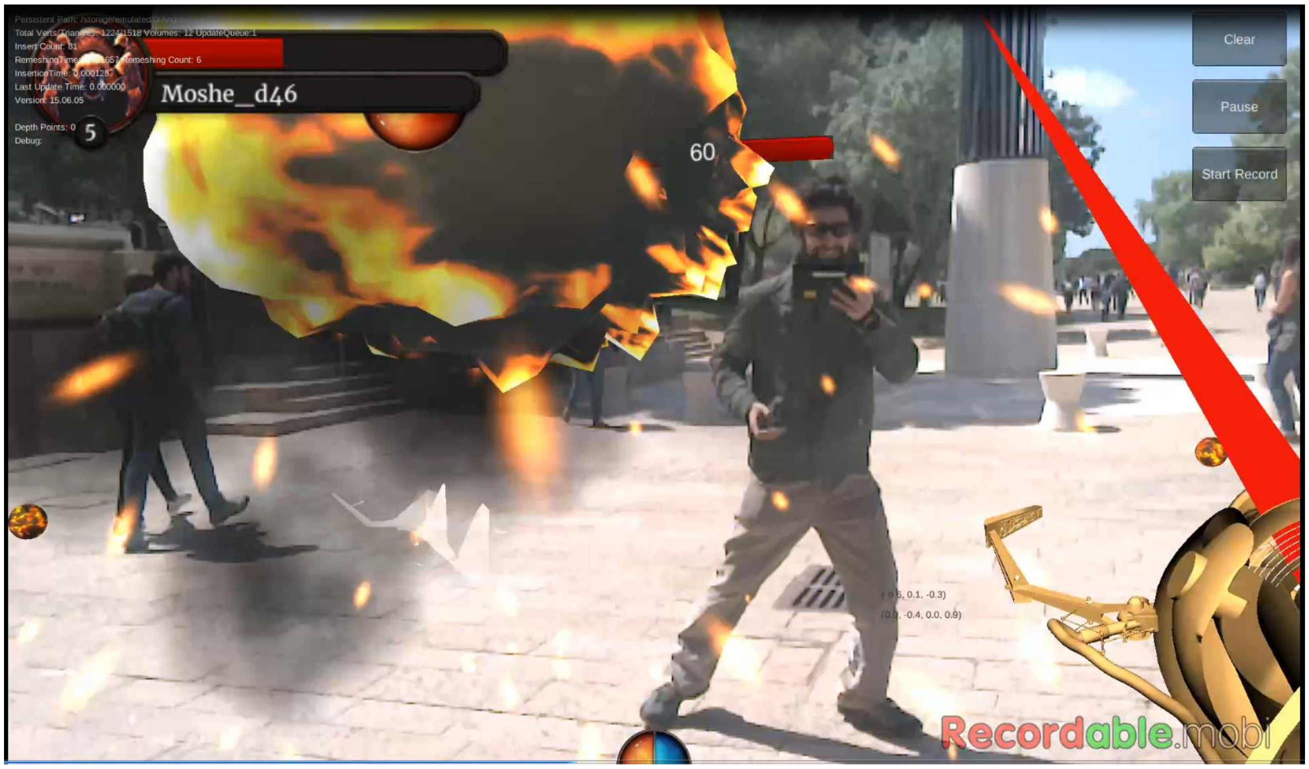
Coop is the key to win!

hurry up, they coming..

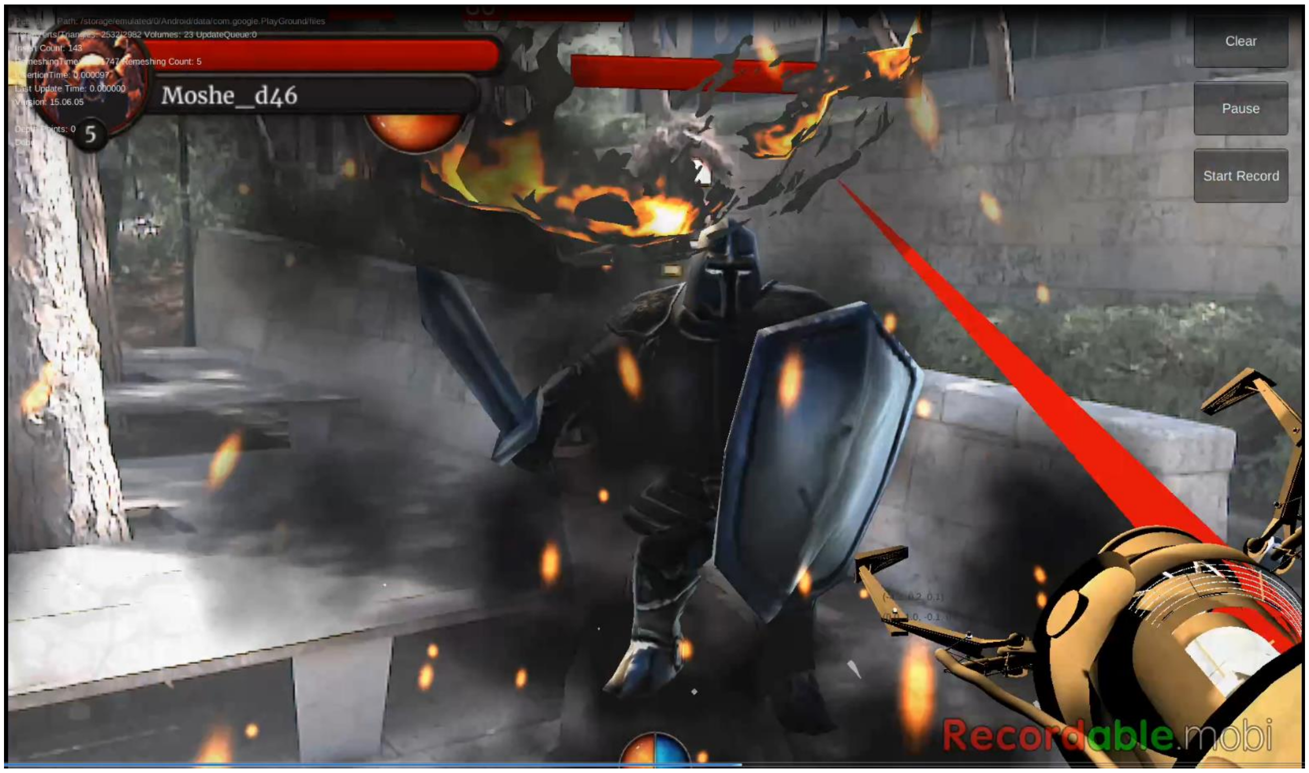
Good luck!



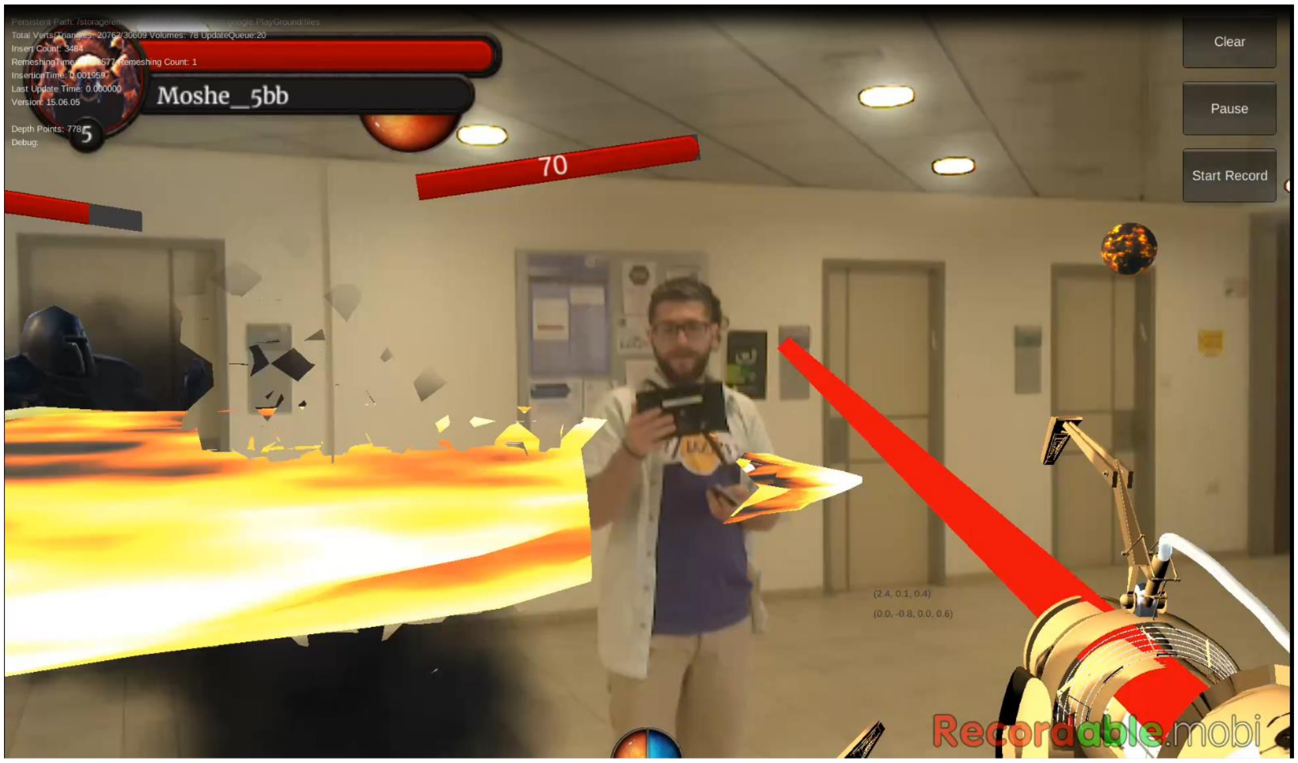
```
Persistent Path: /storage/emulated/0/Android/data/com.google.PlayGround/files
Total Verts/Triangles: 3908/4776 Volumes: 52 UpdateQueue:0
Insert Count: 451
RemeshingTime: [illegible]497 Remeshing Count: 2
InsertionTime: 0.000452
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 0
Debug:
```



```
Persistent Path: /storage/emulated/0/Andro[illegible]
Total Verts/Triangles: 1224/1518 Volumes: 12 UpdateQueue:1
Insert Count: 81
RemeshingTime: [illegible]657 Remeshing Count: 6
InsertionTime: 0.000125
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 0
Debug:
```



```
[illegible] Path: /storage/emulated/0/Android/data/com.google.PlayGround/files
[illegible]rts/Triangles: 2532/2982 Volumes: 23 UpdateQueue:0
In[illegible] Count: 143
[illegible]meshingTime: [illegible]747 Remeshing Count: 5
[illegible]ertionTime: 0.000097
Last Update Time: 0.000000
V[illegible]sion: 15.06.05

Dep[illegible] Points: 0
Deb[illegible]
```



```
Persistent Path: /storage/em[illegible]google.PlayGround/files
Total Verts/Triangles: 20762/30609 Volumes: 78 UpdateQueue:20
Insert Count: 3484
RemeshingTime: [illegible]577 Remeshing Count: 1
InsertionTime: 0.001959
Last Update Time: 0.000000
Version: 15.06.05

Depth Points: 778
Debug:
```