DSP- RWTH
ITAR Team Demo Project
Test Documentation

Created by Rohan Fernandez

Content

1) Introduction & Instructions
2) Implementation
3) Profiling

1) Introduction & Instructions

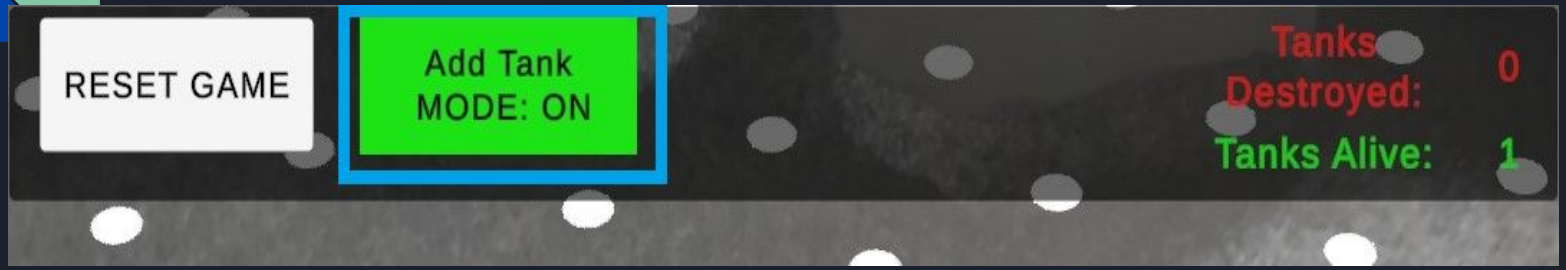
1.1 Introduction and Instructions

- The project is created for DSP-RWTH and built for Android in AR. It is created using the Unity game engine.
- The repository for this project can be found with the link https://github.com/RohanFernandez/DSP_TankAR
- A mobile phone with a minimum version of android 11.0 (API level 30) is recommended.
- Copy the built application file ‘DSP_ITAR_AR_BUILD_RohanFernandez’ onto your phone.
- Install the .apk and an application named DSP_TankAR will be available.
- Click on the application DSP_TankAR to start the application.
- Accept permissions for the usage of the camera if asked for.

1.2 Startup and Surface Detection

- Once started, you will have the camera turned on and a UI panel displayed.
- Move the phone around pointing the camera at surfaces to detect flat horizontal surfaces.
- The application is setup to only detect flat horizontal surfaces.
- These surfaces are indicated as white dots (blue inset).
- There are UI panels that allow you to control/edit the application, like the panel on top (pink inset).

1.3 Adding a tank



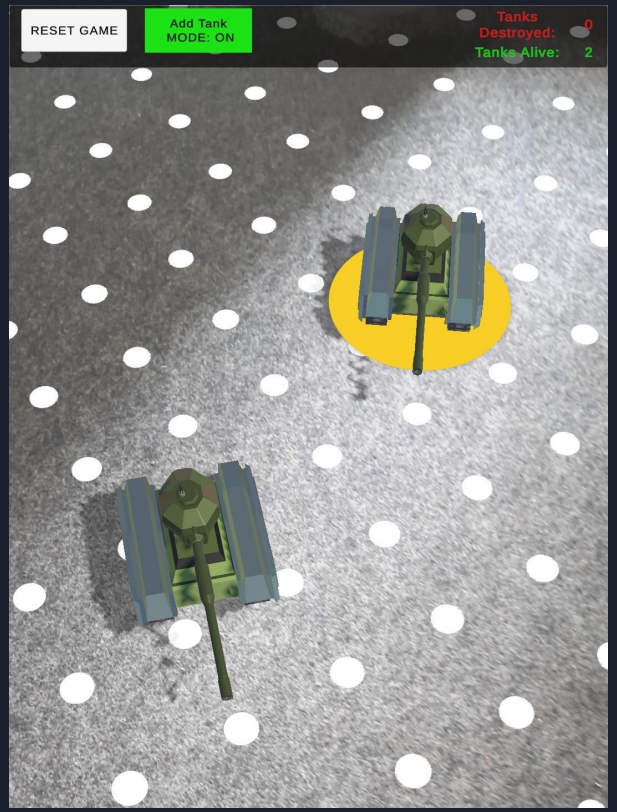
- The top UI panel includes
  - ‘RESET GAME’ button
  - ‘Add Tank MODE: On/Off’ toggle
  - Label to indicate number of tanks destroyed and currently alive
- The game includes an ‘Add Tank’ mode that can be toggled On/Off
- To toggle between the modes click on the toggle (blue inset)
- A Tank can only be added into the scene while this mode is ON

1.4 Adding a tank

- To add a tank into the scene make sure the Add Tank Mode is On.
- Tap on a point on the detected plane to spawn a tank.
- The first tank added will be the ‘selected tank’. Only a single tank will be set as the selected tank.
- The selected tank is indicated with a yellow colored circular base.
- Only the selected tank can be controlled when the ‘Add Tank Mode’ is Off.

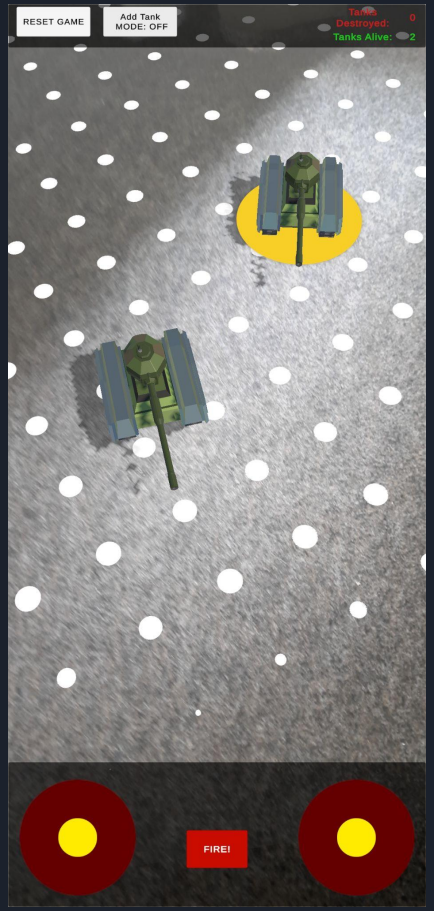
1.5 Selecting a tank

- It is possible to select any tank that is added into the scene.
- To select a tank, tap on the tank you want to select.
- The yellow colored circular base will then be under the current selected tank.
- You can select a tank while the ‘Add tank’ Mode is On or Off.
- Additionally, you can drag a tank along the plane to reposition it.



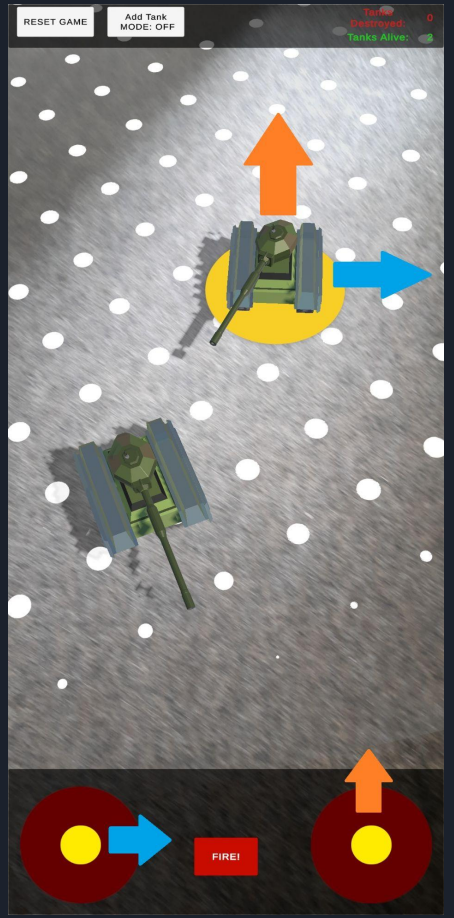
1.6 Controlling a tank

- To control a tank, toggle the ‘Add Tank’ mode Off.
- If a tank is selected, a bottom panel is displayed that enables two joysticks and a Fire button that allows you to control the selected tank.
- The left joystick allows the selected tank to move horizontally along the detected plane.
- The right joystick allows you to rotate the canon on the selected tank along the Y-axis.
- The fire button shoots a rocket from the selected tank.



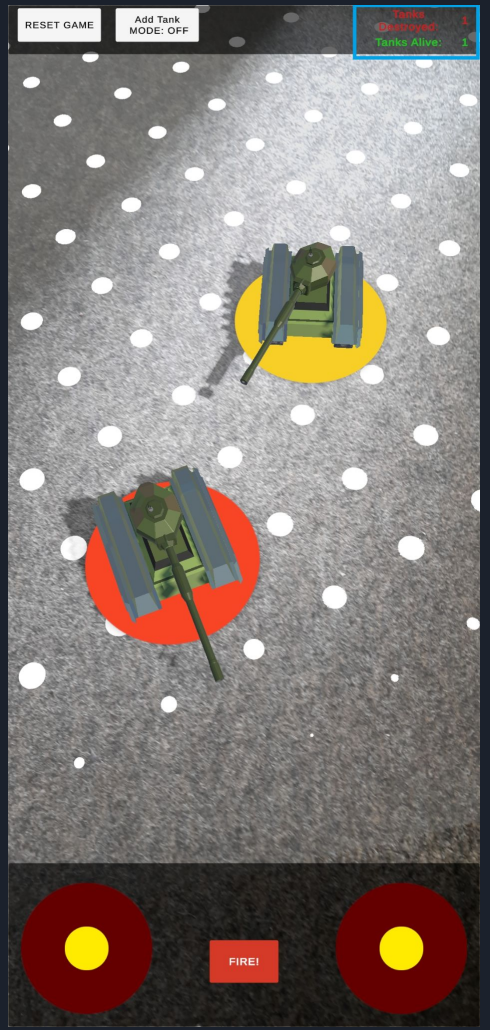
1.7 Joystick Control

- The left joystick moves the tank forward and rotates the tank towards the direction of the joystick (Blue Arrow - Rightwards from the camera).
- The right joystick rotates the canon of the selected tank towards the direction of the right joystick (Orange Arrow - Away from the camera).



1.7 Shooting a tank

- If you shoot another tank, the tank that is hit is ‘destroyed’.
- A ‘destroyed’ tank cannot be selected and is indicated with a red colored circular base.
- A panel at the top right corner (blue inset) includes details of how many tanks are currently alive and destroyed.



1.8 Reset Game

- The top panel also includes a ‘Reset Game’ button (blue inset).
- On clicking this button all tanks currently present in the scene will disappear.
- The number of tanks destroyed and alive will also be reset to 0.
- It allows you to restart the game and set the tanks into the scene again.

2) Implementation

2.1 Implementation

- The project is implemented with a set of managers constructed hierarchically.
- The Game Manager controls the game states and sends events to the managers.

- GAME MANAGER
  - UI Manager
    - Shows/hides all UI based on game states.
  - Input Controller
    - Manages touch input and joystick touch-axis values.
  - Tank Manager
    - Manages instances of tanks.

2.2 Game Manager

- The game manager manages Game states.
- There are 2 game states: Add Tank, Gameplay.
- ‘Add Tank’ State
  - This state allows the user to add tanks into the scene.
  - The tank cannot be controlled in this state.
- ‘Gameplay’ State
  - This state allows the user to control the selected tank.
  - A tank cannot be added into the scene.
  - Prevents accidentally touching the screen and adding a tank.

2.3 UI Manager

- The UI manager is used to manage Showing or Hiding UI components based on game states.
- When a game state is changed, the Game Manager sends an event indicating the changed event.
- The information on the top UI panel i.e. Tanks alive/destroyed are updated.
- The bottom UI panel is shown or hidden based on if the current game state is a Gameplay state.

2.4 Input Controller

- This manager checks for touch events on the screen, example: Finger down/move/up events.
- It checks if the touch was on screen to add a tank, to select a tank, right and left joystick movement.
- The values recorded by these events are sent to other managers, for ex: to add a tank at a position.
- It manages multi-touch input values for the 2 joysticks.
- It also manages the click on the Fire button because Unity UI components do not invoke events in cases of multi-touch.

2.5 Tank Manager: Tank and Rocket instantiation

- The Tank Manager has the responsibility to efficiently manage the instances of tanks and rockets.
- To avoid instantiating a tank or rocket each time it is needed, the concept of an Object Pool for each type is used.
- A past project I developed, allows for objects to be re-used which I implemented here https://github.com/RohanFernandez/ObjectPoolingSystem
- When the game is reset, the instances of tanks are deactivated and returned to the pool.
- Additionally, a default number of objects for that type are instantiated on start of the game and are deactivated.
- This prepares for its eventual usage in the game.

2.6 Tank Manager: Object Pooling

- At any instance, only a certain number of rockets can be present.
- Instead of instantiating them, they are reused preventing unnecessary repeated memory allocations during gameplay.
- A similar example is seen when loading a scene. A loading panel is displayed covering up the lag that can be seen on allocation of the new scene in memory.
- When a Tank or a rocket is not needed it is deactivated instead of deleted.
- This prevents the Garbage Collector from being called to manage the newly deallocated gap within the memory.

2.7 Tank Manager: Reuse of objects

- From the image it can be seen that 6 tanks are created by default at the start of the game.
- During the game 3 are requested and hence are activated from the 6 pooled.
- Eventual need of tanks are taken from the remaining deactivated but pooled tanks and returned after its usage.
- A similar concept is conducted with the rockets.

2.8 Tank Manager: Tank Control

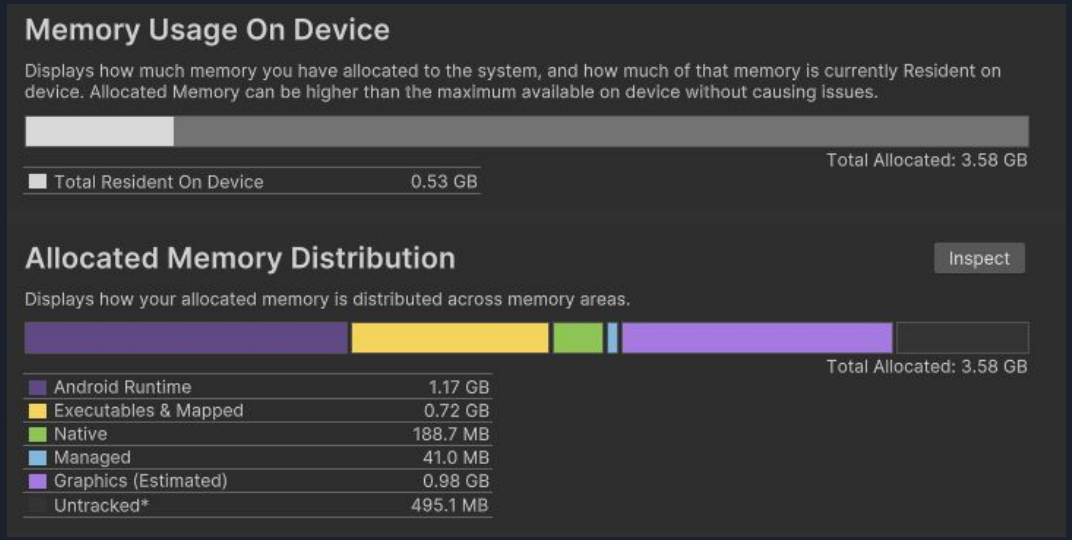
- The Tank Manager has a single instance of a Tank that it sets as the Current Selected Tank.
- All touch events from the Input Controller are sent to the Tank Manager, indicating to create a tank at a location.
- Also events with regard to control i.e. movement, canon rotation, fire are intended for the Current Selected Tank.
- The direction controls from the joystick are with respect to the camera forward direction at that instance.
- When a rocket hits a tank, the tank invokes a callback set by the tank manager.

3) Profiling

3.1 Profiling

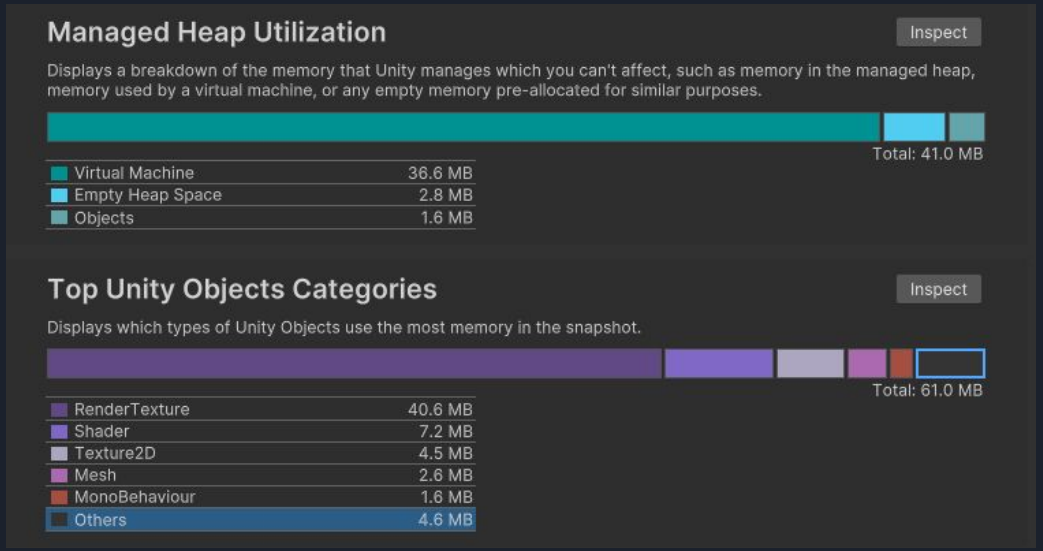
- The following tests are done to profile the application
  - A memory snapshot of the application
  - Profiling FPS and CPU function execution times
- The specifications of the device used for development and profiling are as follows:
  - Samsung Galaxy M31
  - Android Version 12
  - 5.8 GB RAM
  - Octa-core (4x2.3 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53)

3.2 Total Memory Allocated: 20 Tanks



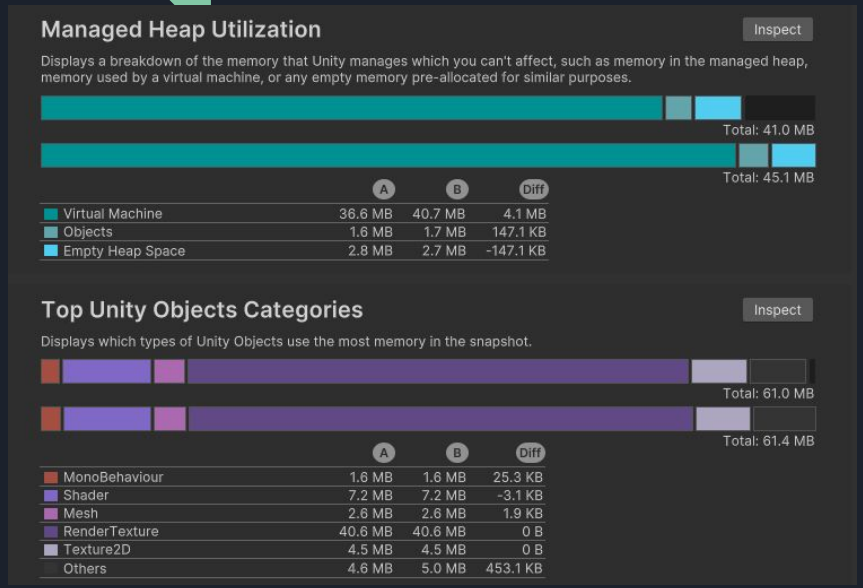
- Total Allocated : Available memory on device
- Resident : Memory of the application in physical memory
- Managed = 41 MB, Managed Heap Memory

3.3 Profiling memory with 20 Tanks



- A total of 61 MB is used
- Heap = 41 MB
  - Virtual Machine : Managed Type Metadata
  - Objects : MonoBehaviour Game objects referenced
  - RenderTexture: Used to draw with depth related information

3.4 Comparison: 20 Tanks Active vs 20 Tanks Active (30 Tanks in Pool total)



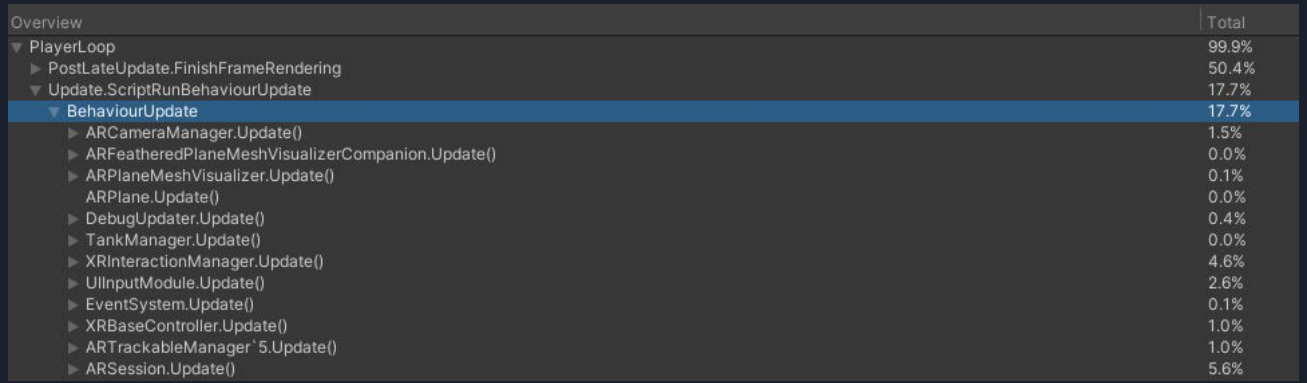
- A : 20 Tanks Active in Scene
- B : 20 Tanks Active in Scene (10 Deactivated in the pool)

- On the Heap, B takes +147.1 KB to reference the extra objects.
- A small amount to reference the pooled objects.

3.5 CPU Profiling

In a development build with 20 tanks

- Avg. FPS : 13.3 FPS.
- From the Profiler, highest time is taken for rendering.
- In the Behaviour Updates : the AR Manager takes the highest amount of time followed by the XR interaction and then the Input Module.



Overview	Total
▼ PlayerLoop	99.9%
▶ PostLateUpdate.FinishFrameRendering	50.4%
▼ Update.ScriptRunBehaviourUpdate	17.7%
▼ BehaviourUpdate	17.7%
▶ ARCameraManager.Update()	1.5%
▶ ARFeatheredPlaneMeshVisualizerCompanion.Update()	0.0%
▶ ARPlaneMeshVisualizer.Update()	0.1%
ARPlane.Update()	0.0%
▶ DebugUpdater.Update()	0.4%
▶ TankManager.Update()	0.0%
▶ XRInteractionManager.Update()	4.6%
▶ UIInputModule.Update()	2.6%
▶ EventSystem.Update()	0.1%
▶ XRBaseController.Update()	1.0%
▶ ARTrackableManager`5.Update()	1.0%
▶ ARSession.Update()	5.6%

3.6 Instantiating instead of Pooling

- To test the difference, the rockets are instantiated and destroyed continuously each time instead of reusing them from the pool.
- From the profiler spikes can be seen in the ‘GC Allocated in frame’, related to the garbage collector recuperating on the destroyed memory. This causes a lag in FPS to 6.3 FPS.
- Similar can be seen on ‘Reset Game’ with the tanks.