SPACE GUARDIANS’ MULTIPLAYER

**DEVELOPMENT LOG**

This is the development log for the Space Guardians multiplayer (client/server) setup using the MP Tanks / CBS (Cross-Platform Backend Solution) as a foundation for the new game.

The MP Tanks project offers a foundation for the multiplayer arena style systems which uses Unity’s Netcode OR **Photon PUN2** (we used Photon) for its matchmaking services combined with Microsoft’s Azure PlayFab dedicated server back-end for user authentication, inventory, user management and databases.

The CBS (Cross-Platform Backend Solution) package provides a selection of easy-to-follow tools to manage inventory, itemisation, events, giveaways, battle passes and so much more that are provided as part of the Azure PlayFab dedicated server hosting.

**-- IMPORTANT–**

**This document is important and discloses important information regarding the development and maintenance of core systems in the project. DO NOT disclose this to ANYONE. If you have access to this document it is intended for your eyes only and should not be replicated or shared with anyone without express consent from its author, Gareth James, DEFVERSE Founder and creator of the Space Guardians game.**

* **PHOTON SETTINGS**  
  Open in UNITY: Window > Photon Unity Networking > PUN Wizard  
  **APP ID:** a71d3a81-aef9-4327-9ba7-08b87f8995ad  
  **Photon PUN2 Services:** <https://www.photonengine.com/pun>
* **PLAYFAB SETTINGS**Use the Playfab Editor Extension package!The window should show in the same panel as the Unity inspector. Open in UNITY > Window > Playfab > Editor Extensions  
  **APP ID:** 5F8EA  
  **Microsoft Azure PlayFab Dedicated Server:** <https://playfab.com/>
* **MP TANKS BASE GAME**  
  Open in UNITY > Window > Tanks Multiplayer > Network Setup. Follow the PUN2 setup if you want to replicate our set up   
    
  **MP Tanks Project Unity Asset Store:** <https://assetstore.unity.com/packages/templates/tutorials/tanks-multiplayer-netcode-photon-69172>
* **CBS (Cross-Platform Backend Solution)**  
  Open in UNITY > CBS > Configurator. There are several steps to follow so please read the documentation and follow it step by step. It’s complicated so don’t skip steps.  
   **CBSS (Cross Platform Backend Solution) Unity Asset Store:** <https://assetstore.unity.com/packages/tools/game-toolkits/cbs-cross-platform-backend-solution-playfab-200638>

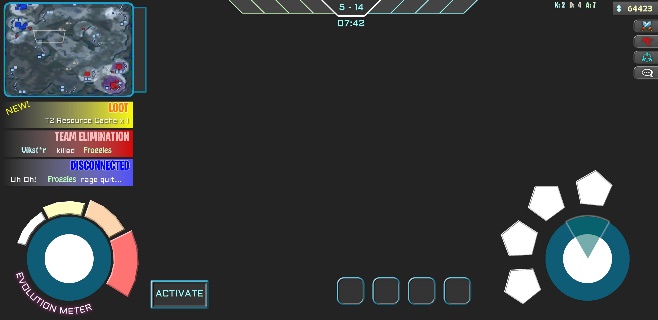
**-- TO DO LIST –**

1. Progress: At 4.1 in the documentation. Started editing the movement of the spaceship but it’s not working as intended, need to fix the following:  
   1. Spaceship should NOT accelerate faster while going forward/strafing at the same time. Need to add a multiplier of 0.7f to restrict speed while pressing strafe and accelerate/decelerate.
   2. Look into using rigidbody.velocity again instead o fusing translate because velocity adds a acceleration/deceleration rate which looks really cool for spaceship style movement. May need to tweak the acceleration rates (and/or cap the maximum velocity). Check what affect gravity has on the rigidbody and if we can add our own drag factor.
   3. Building upon the current controller in the Player.RotateShip2() function, think about adding more modifiers for move and strafe. Maybe with some drift velocity on x/z that dampen over deltaTime. This will offer the ability to customize ships with a more spacelike feel.
   4. Need to fix the right thumbstick so it represents current rotation and input (through mouse + touch). Also need to set up the thumb stick so that it works on mobile through touch. On second thoughts, on PC it isn’t shown anyway, only in Unity Editor so might not be all that bad, but do need to ensure the current (or chosen) controller works seamlessly on both devices.
2. Continue from the TanksMultiplayer.pdf documentation tomorrow. Was working through the documentation one more time just to ensure I understand everything that the tanks demo offers and then I will move on to the CBS setup which will lead me through the setup of the Azure Playfab services so we can test things like our own item store, authentication, and inventory management. TANKS DEMO DOCS FIRST > CBS documentation next.
3. Basic things to do once the documentation have been read:  
   1. Add a new tank (how easy is it, what we gotta change?) - DONE  
      Things to consider:
      1. Tanks change colour when they’re assigned a team, override this.  
         COMPLETED: No need to override, use a different object to represent the team colour, perhaps an ellipse or just a cube indicator or something.
      2. Turret not needed on spaceship, see how to achieve this
      3. Controls for spaceship are different need to move differently to the tank setup so we need to fix this too
         1. Left stick – strafe left right / forward + back movement
         2. Right stick rotate
   2. Change the tanks to spaceships
   3. Disable the turret rotations, they’re not needed.
      1. Weapons auto fire, default auto cannons will fire directly in front of the ship (should not be gimballed). Must always have ONE weapon firing directly ahead of the ship? Perhaps make all auto cannons directional to prevent any issues?
   4. Add a new scene (or duplicate the TDM one) and change the environment. Fiddle around with the navmesh stuff and see if you can extend the AI to roam further than it currently does now.
   5. Implement SG enemies and see if we can have them move around the same way as the tank bots do at present.
   6. Implement CBS and try creating a store and adding the ships to the store.
   7. Use CBS to create a leader board and save high scores from the TDM mode.
      1. Each player gets points based on K/D/A + team score
      2. Each players score gets added to their current leader board score so that the leader boards can update with the most productive players
      3. (Kills / Deaths) gives KD average + (Wins / Losses) gives match average + (MVP earned / Games Played) gives MVP average = TOTAL SCORE

-- v0.1.4 –

* **SG\_Weapon.cs (Scriptable Object Template):** In the single player demo VertShooter project, I used the Weapons script to create scriptable objects. Scriptable Objects allow us a way to create a template that all weapons can use. I have edited the original Weapon.cs script from the single player demo significantly to work with this project. I have also formatted it using the Odin Inspector Scripting to organize it neatly and make it a lot more user friendly
* **First Weapon (Scriptable Object):** So as a result of adopting the SG\_Weapon.cs script, I have made the first weapon template which I have named DEW-2 (Directed Energy Weapon, and 2 because of the number of simultaneous lasers it can fire). This is dual laser turret that fires red laser beams.   
    
  **TEMPLATE ID: 103001\_Laser\_DEW-2**  
    
  All weapons templates are to be stored in the /Assets/Space Guardians/Weapons/ directory.
* **NEW PACKAGE – Odin Inspector / Validator / Serializer:** I bought this package a while back and never got around to installing it. This package helps create extremely customizable Editor scripts that help simplify and tidy up the default Unity Inspector. The validator helps check for builds errors, missing files, null exceptions and a ton more. General all-round upgrade and development lifeline. Began using the Odin Inspector to create the Weapon.cs layout. It’s pretty neat.
* **DEV VIDEO:** Made a quick 2 minute dev video outlining the progress on the Weapon.cs script and a quick overview of the Inspector Panel I designed. This video can be found here:   
    
  **E:\1 - MUST KEEP\1. - Games Dev\Space Guardians\Videos\progress vids\1. SG MultiPlayer Dev Progress Videos\20230314-SGMP-WeaponsTemplate+CodePreview-NOT FOR PUBLIC.mp4**
* **NEW SCRIPT (SG\_Powerup\_RandomWeapon):** is a script that will be attached to any random weapon power up which when collected, triggers a random weapon to be selected from the players weapons pool. If they have an available weapons slot in the Player.WeaponsReady List (to be implemented in player.cs) then the randomly chosen weapon will be assigned to the first available on screen button, and added to the WeaponsReady list so it can be activated when the player chooses by pressing the associated Weapon Button on screen.
* **NEW NAMING CONVENTION:** All scripts that are to be added to the project here onwards which are created for Space Guardians specifically or written solely by me will have the **SG\_** prefix so I can easily identify scripts that belong to the project. These scripts are held in the MPTanks/Scripts folder so this will allow me to keep track of what script was written for the Space Guardians project specifically to avoid confusion later.
* **Sdsd:** sdsd

-- v0.1.3 –

* **New Branch (RemoveTurret+Controls):** This new branch will focus on removing all the turret functions from the player.cs script altogether and coding the player control system from scratch to see if it’s better than the current approach.
* **Mobile UI (In-Play):** In-play UI concept design completed. Followed typical MOBA layout, integrated evolution meter into the left control stick with independent activator button to the lower right. Subject to change. First concept. NOT final art, only layout design created for UI artist to use as reference.  
    
  Graphical user interface, application

  Description automatically generated
* **PlayerCam + UIBillboard.cs:** Attached a playerCam to the ShipVulcan prefab. ShipVulcan will eventually just be PlayerShip and will have ALL models attached to it which can be individually activated with a simple ACTIVE flag from a list of available ship models (is this a good idea? Research any associated memory costs / cheaters unlocking it (perhaps encrypt the list + obfuscate the code))  
    
  The UIBillboard has been modified to check for this camera using its “PlayerCamera” tag, and if it cannot find it, it will default to the MainCamera which hasn’t been removed as it has the Audio Listener attached to it. Removed the AudioListener from the PlayerCam.
* **Disabled ReceiptValidator.cs:** To allow the program to be built I have had to disable the ReceiptValidator script. We can enable this later once we link the app with Google / Apple. Just without these App IDs, it’s throwing a null exception which prevents the program from being run. I have simply commented out the following line (88) in the Initialize() function to achieve this.  
    
  **localValidator = new CrossPlatformValidator(GooglePlayTangle.Data(), AppleTangle.Data(), Application.identifier);**
* **Player Controls (Player.cs):** I have amended the controls so that the player ship now moves and strafes off the left stick + WASD, and the rotation is handled by the right stick + mouse position. I have attempted to remove the AimIndicator from the mouse position but it’s proving difficult.  
    
  Created a small sample video showcasing the first draft movement style. Comes with various sliders for rotation and movement including accel/decel modifiers and dampeners. A lot of customisations can be had here. May release this to BETA TESTERS to fiddle with and get some feedback, would have to make a test menu though.   
    
  **Video**: Control Example 1 - Arcade Style - Optional Drag and Drift - Very Mouse Intensive (lots of lifting and readjusting depending on sensitivity).mp4  
    
  **Location**: E:\1 - MUST KEEP\1. - Games Dev\Space Guardians\Videos\progress vids\080323 - SG multiplayer - First control sample  
    
  **RECOMMENDED SETTINGS (need a lot of tuning though)**  
  Text

  Description automatically generated
* **RotateShip() and RotateShip2() Functions:** Made 2 for now, first one attempts to rotate the ship to the thumbstick position. Handy for the mobile, terrible for the PC. Would need work to stop it being able rotate 180 by moving the mouse position behind the ship. This makes an awkward movement scenario when moving the mouse north of the ship again. Also, the thumbstick representation would need to be modified to represent the look direction accurately.  
    
  RotateShip2() is the current version I’ve been working on, this uses:   
    
    
    
  to get the value of the movement in a float. I’ve used a value to clamp the max input and added several other variables that can be used to manipulate and configure the rotation and other player movement attributes. This needs more work as it seems a little clunky.

Text

Description automatically generated

* **RotateShip3() Function:** Created a new rotatship function that uses only keyboard input. WASD and Left / Right arrow to control the rotation. Works as intended. Next step is to apply roll and test mobile stick controls against the current control system. Might have to use a conditional compilation using #if statements to work only when using mobile devices. Rename this to something more relevant like RotateShipKeyboard() later.
* **ISSUE: Support Request FLOBUK:** Request sent to Florian for help with the mobile control inputs. They’re very small but also the main issue is where the sticks are applying the movement/rotation. It seems they might be doing it in UIJoystick.cs but I have sent a request and paid for 1 year support to help clarify.
* **Weapons NFT Headers:** Designs for the weapons NFT header categories have been finalised by Thanh. Attached are the previews. .psd’s have been stored here: **\DefVerse\1. Master Assets - ART\DEFVERSE ASSETS - THANH BUI\1. MASTER FILES\NFT Borders\Weapon NFTs**  
    
  

-- v0.1.2 –

* **Code:** Had to roll back the project to the initial setup stage before CBS was implemented as there were issues with some of the MPT sounds. They weren’t linked any more, couldn’t figure out to remedy it so just rolled back for now. All is working again.
* **New Tank:** Successfully added new simple tank to the shop screen.
* **Shop Vertical Scrolling List:** Added a vertically scrolling list that can contain lots of shop entries. Just tested the functionality, this will be removed when we start using the CBS scrolling battle / item store options later.
* **Tank Renderers:** When you want to an object to represent the team colours for the tanks / spaceships, you need to add their primitive shape to the Renderers list which is located on the Player script? 🡨 check this, not sure. Only objects added to the Renderer list will be turned into the corresponding team colours. For SG we’ll need to just have a semi-circle or something similar below the player object so that we don’t mess with the models themselves.  
    
  UPDATE: Added a sprite image below player now that shows a default yellow colour. Replace this later to an ellipse primitive that can be rendered in team colours. Won’t work on the Quad for some reason as it seems to reorientate it too. Perhaps a misconfiguration with the turret rotation I was messing with earlier. Look into this again later.
* **Spaceship Model Added:** Added the first SG spaceship Robocorps Vulcan to the game. Had to add point light to the prefab. Duplicated the tank prefab and changed the model. In the process of editing the movement properties and removing (repurposing) the turret rotation/look method.
* **Arena Design:** Decided that we need to offer different skill types such as AOE, targeted AOE, Conal Attacks, and directed fire for PvP mode. We need to do this to ensure skill plays a part in the success of the player while playing PvP. Cards will be selected via power ups, activated nu skill buttons, and directed by the user where they will begin to fire until their timer elapses.

-- v0.1.1 –

* A backup of the MP Tanks base project has been saved here: **U:\Space Guardians Backups\** **Multiplayer Project - SG - before CBS setup 0.1.1.rar**This is a backup with just basic PlayFab install, account linked, and the CBS (Cross Server Backend Solution) unity package has been added also but has not been configured yet.

-- v0.1.0 –

* Initial setup of the project including the addition of the MP Tanks, Photon PUN2, and CBS packages.
* **GIT - NEW BRANCH:** Created a new BRANCH “CBS-Setup”. This is to contain all modifications to the MP Tanks codebase + the integration of Cross-Server Backend Services package until it’s all up and running and working as intended.
* **NEW SCRIPT:** Added a script called JustTesting.cs which is located in the /TanksMultiplayer/ folder. This is just for my test scripts as I play around with some of the code. This is to be removed from live builds.