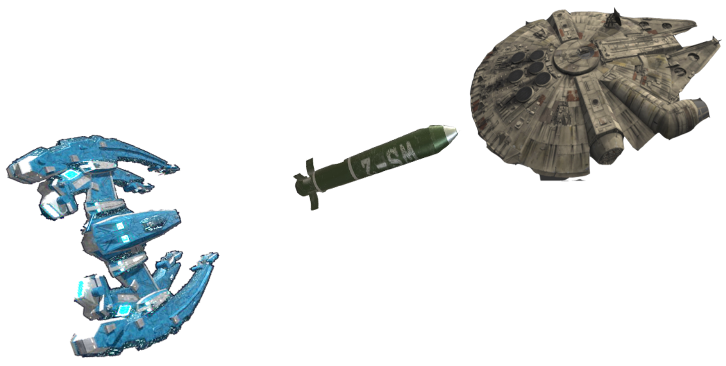
Computer Graphics Project
BE in space
Game Project

Students: Basel Zarka
Elie Abboud
Advisor: Dr. Roi Poranne

Game Description:

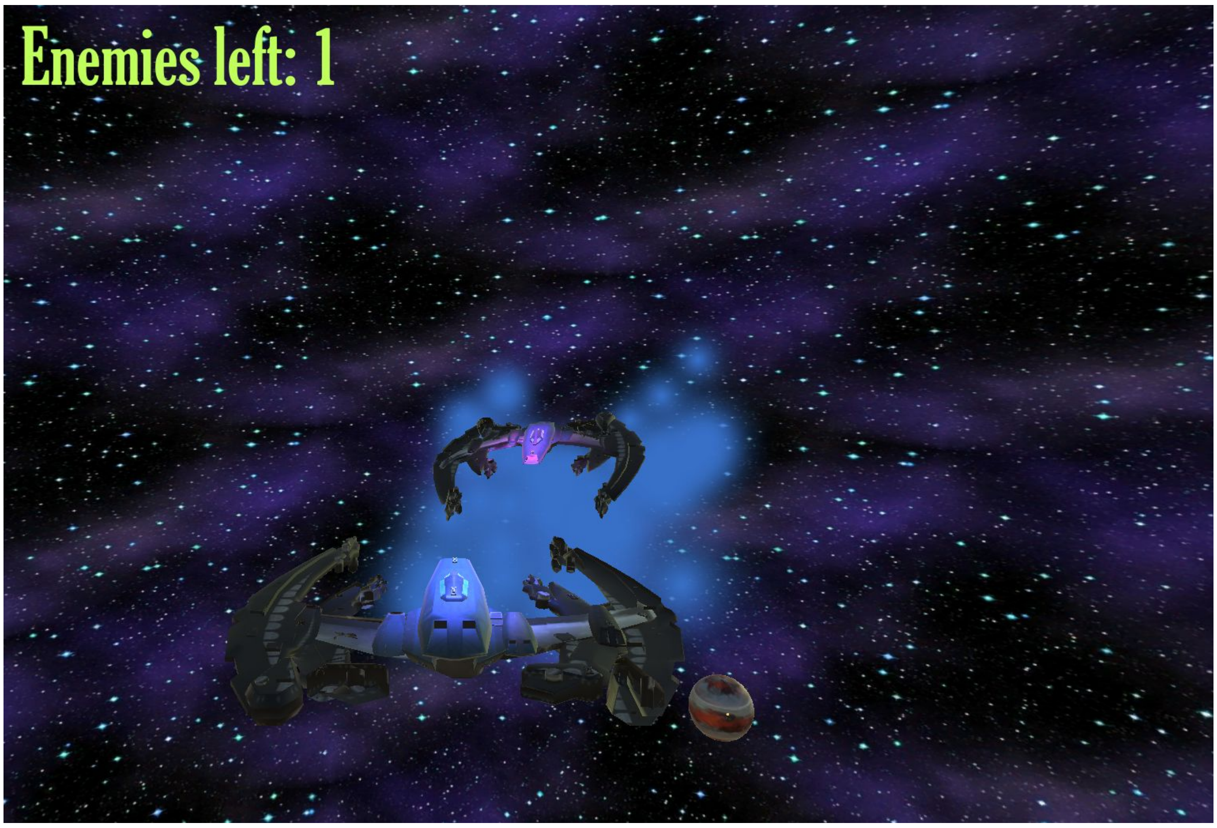


Within the game, the user controls a spaceship. The goal is to take full control of the loaded galaxy. Other planets will attempt to stop you by sending out their most powerful forces (Enemies). As a player you will have to face them all to guarantee your continued reign.
Optionally, you can choose to play with a friend (LAN Multiplayer). Choosing this option will instantiate another spaceship. The other player should be a guest on your local IP.
All game is handling in 3D objects, including moving and rotations.
The game’s world is a sphere, which can be navigated from within. The players will not be able to escape this sphere, doing so will result in a gameover. All movements, translations and rotations are done in a 3D world space.
The enemies that try to kill you are operating on AI movement so they always follow you intelligently and in a unique way, players will destroy enemies by shooting rockets.

Multiplayer:

This is one of the main features of the game.
Up to two players may join on the same network, and they will be loaded with their own player prefabs.

- **<u>Distinction:</u>** Each one having their own distinct color as shown in figure MP.0:

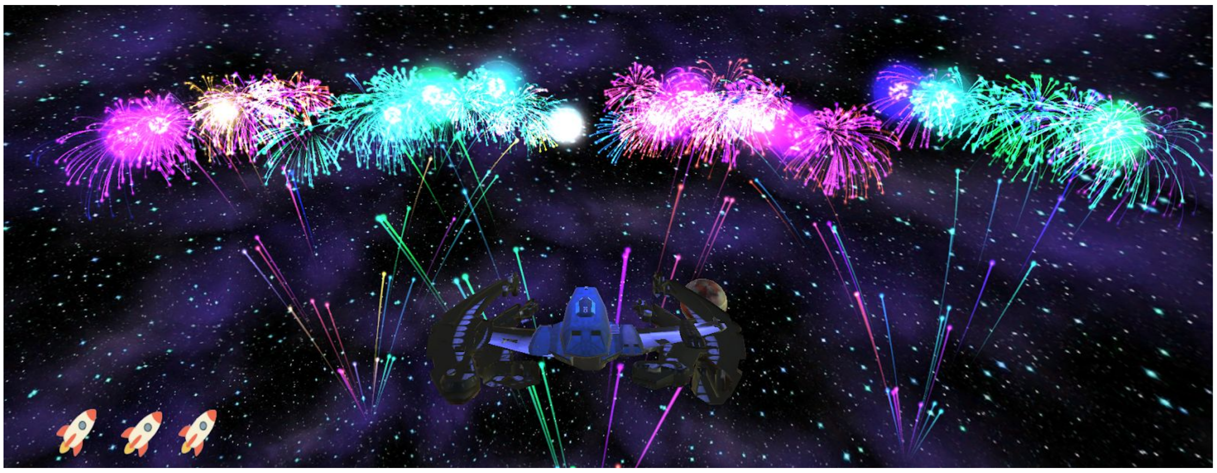


MP.0 - Can see both players, each with their own color.

- **<u>Co-Op:</u>** The multiplayer is a co-op mode, thus the players need to collaborate in order to reach the final level and complete the game.
- **<u>Powerups and Score:</u>** The power-ups (AKA, gifts - described in a different section) are shared by both parties of the game. Thus, when one player collects a power-up, the other immediately benefits from it as well. The score is shared by all parties too, therefore it was decided to be named “COMBINED SCORE”, as shown in figure MP.1:

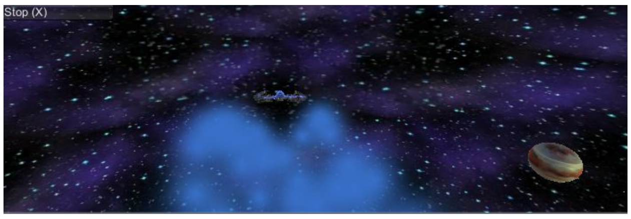
MP.1 - Combined Score GUI

- **<u>GameOver:</u>** When one of the players die, it's game-over for all. And the game will have to be restarted, so you better watch each other's backs!
- **<u>Win:</u>** Winning is also celebrated by all parties. When the game is won a special effect of celebration is shown and the players can roam the world still. However, they can still die if for example they hit an asteroid or attempt to leave the game. The winning effect can be seen as shown in figure MP.2:



MP.2 - Combined Score GUI

Player:

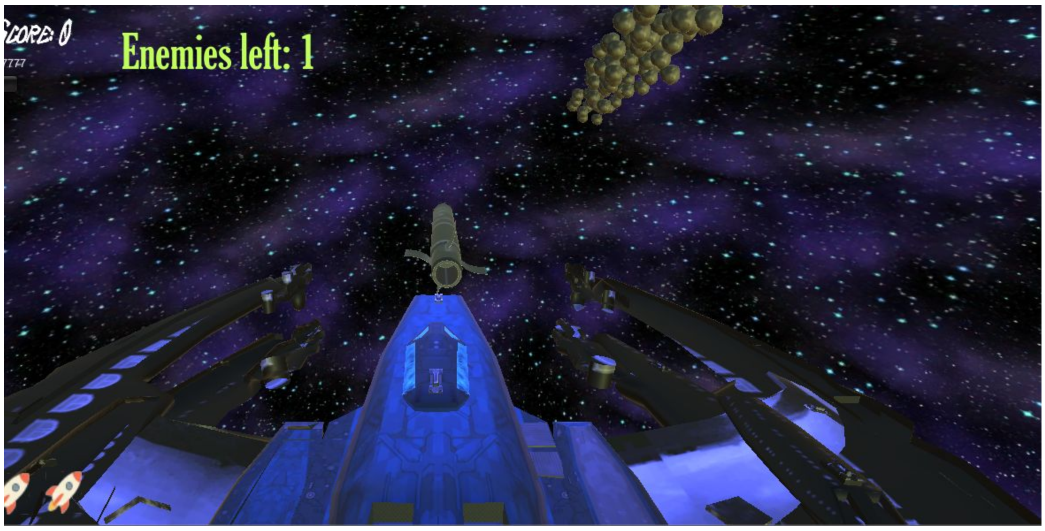


Above is a picture of player spaceship, we handle moving by checking frame by frame which key the user clicked; we also check rotation by checking mouse movements, players have blue tail as an added effect to spaceship “fuel”. As a bonus, it helps players recognize each other from a distance in a vast 3D world.

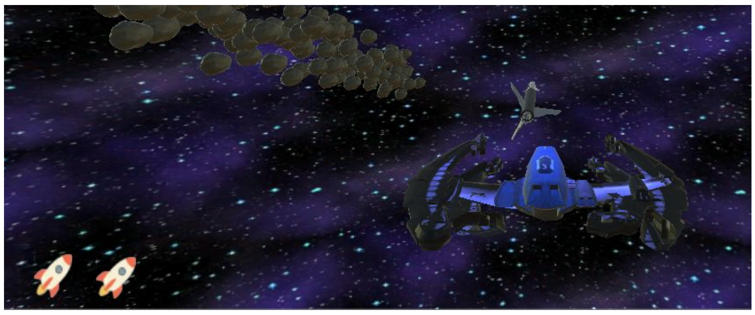
Shooting Shots:

Players can shoot two kinds of rockets:

- **Regular rocket:** Players shoot this rocket beneath the player when Left Click or Ctrl is held down, player have indefinite amount of rockets.

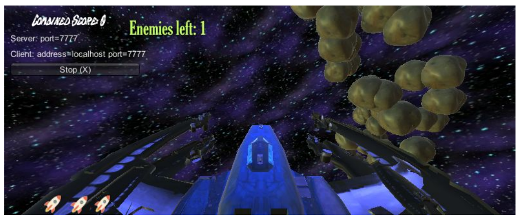


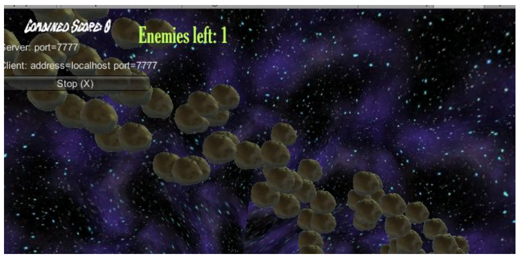
- **Master rocket:** shot by hitting the Mouse Right Click or Alt Key, players have three master rockets reloaded every 10 seconds and lose the ability to shoot them after 7 seconds, master rockets are more powerful and destroy multiple elements in a radius. It'll destroy every element that is within a 20 pixel radius, also if it doesn't hit any element it'll destroy every element with 10 pixel radius.

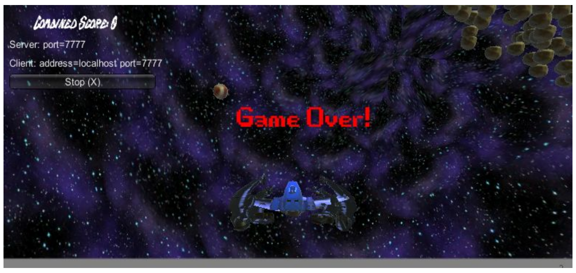


Player Cameras:

Player has three cameras as shown below:

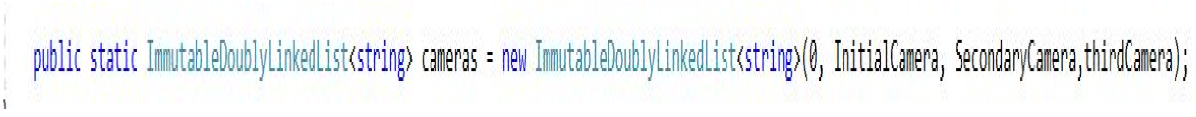






Player can choose to switch between cameras any time by hitting the "Tab" key.

Cameras can easily be added into the game pragmatically, all that's needed to do is add the camera in the Unity Engine and then add its identifying "tag" in the following line:



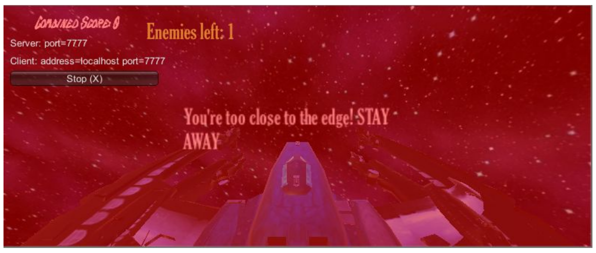
```
public static ImmutableDoublyLinkedList<string> cameras = new ImmutableDoublyLinkedList<string>(0, InitialCamera, SecondaryCamera,thirdCamera);
```

where “InitialCamera”, “SecondaryCamera” and “thirdCamera” are strings that identify a certain camera.

Note: ImmutableDoublyLinkedList is our own custom implementation using C#’s built in List Data structure.

Getting closer to boundary:

Players can’t get outside of the spherical background, otherwise they’ll die, so to inform them that they’re close to the edge a warning alert (flashing screen) with a red color and an endangered sound effect can be heard.

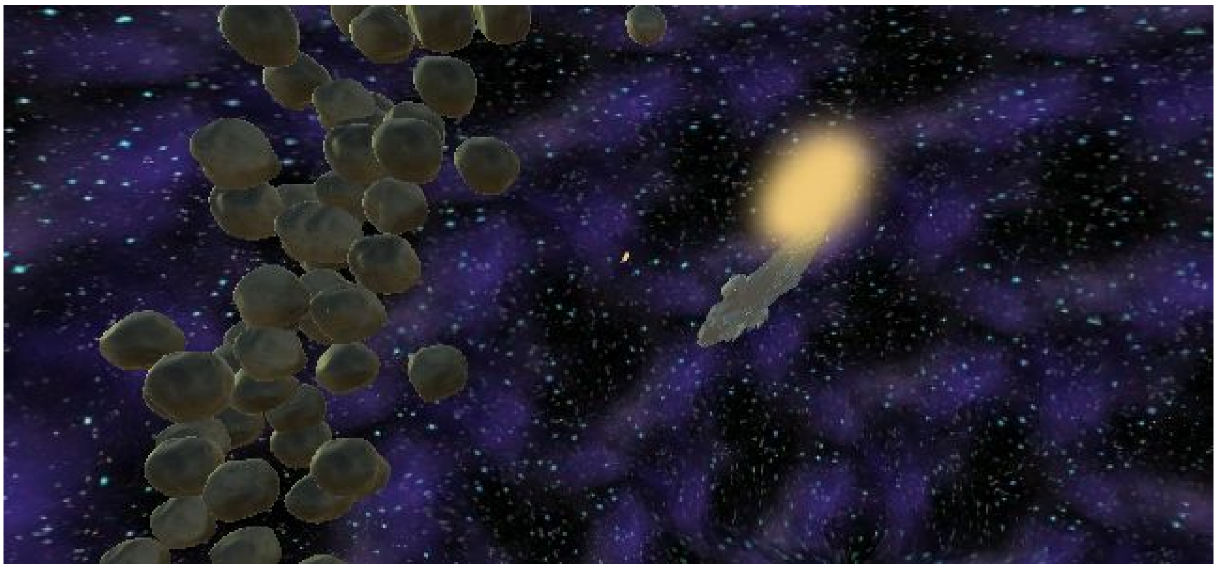


Server: port=7777

Client: address=localhost port=7777

You're too close to the edge! STAY AWAY

Enemies:



There is a single “mesh model” representing the enemy, it also has a tail colored yellow to make players recognize it from a distance. However, with each enemy spawned they have their own distinct attack strategy. In fact, they change their strategy as time goes on.
Enemies have 4 distinct movements and they are chosen depending either on Probability or on how long that particular enemy has been alive.
As a side note, enemies can pause the attack strategy in action to do something specific and then get back to their original attack plan.

The attack strategies are listed below:

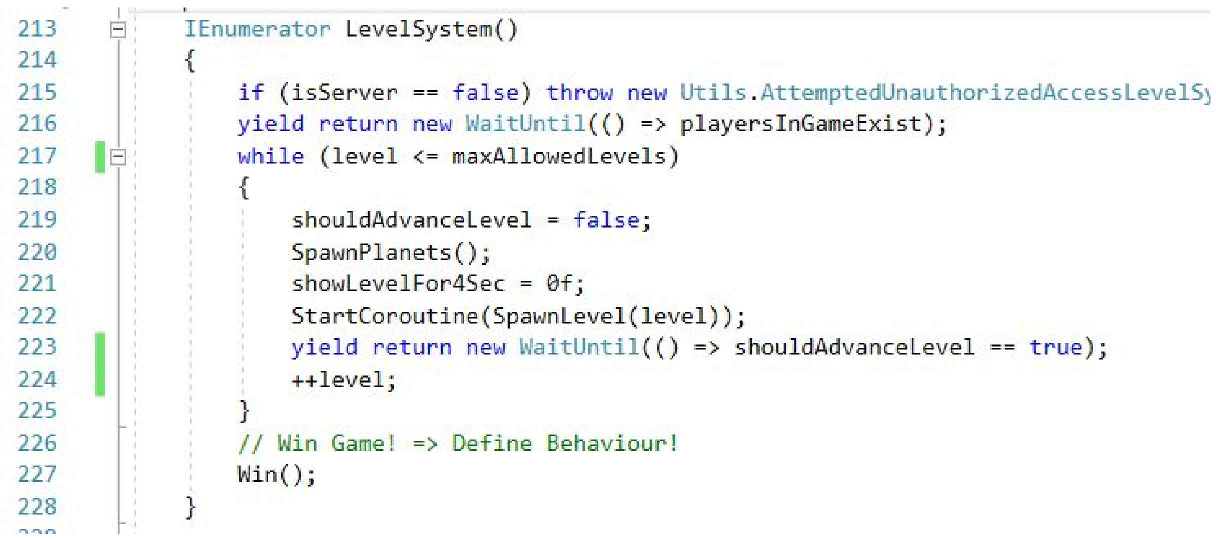
- Enemies that **steer** from other enemies: These enemies are spawned with 50% chance. Thus, approximately half of the enemies will be “steerers”. Once the enemies get too close to one another, they will steer away from each other and then proceed their attack onto the player. This behaviour encourages diversity in their movements and confuses the player. Additionally, the attack will become more dangerous to the player, as it creates a “double attack” from two different sides.
- **Straight Ahead Attack:** or **“Sucuide Attack”** as we like to call it. These enemies attack the player head on, without any special movement. This type of attack is meant to stress out the player to move more.

- **Circulators:** These enemies will move almost in a "Circular Motion" towards the player. They will move in an advancing circle making them harder to aim for due to their rapid changing movement. Once they get close enough to the player, they change to a "Sucuide Attack" described in the previous section.
- **Tricksters:** This is an attack shared by all enemies. When an enemy has been spawned for some time (40 seconds to 90 seconds) they will try to defeat the player in a different way. They will go for the player's current "Blind-Spot" (behind the player) and then proceed with their original attack from there.
- **Random Movers:** This isn't exactly an attack strategy, but it's there to allow for a more diverse and a "realer" feel of the game. This attack will happen occasionally, approximately 20% of enemies are spawned with this "attack". Once they reach their random location, they switch back to their original attack.

Note #1: Enemies may choose to switch their attack plans (with certain probabilities) under certain conditions. For instance, if they've been doing one attack for too long.
Note #2: During Multiplayer, each enemy will choose to chase a player at a 50% chance.

Leveling System:

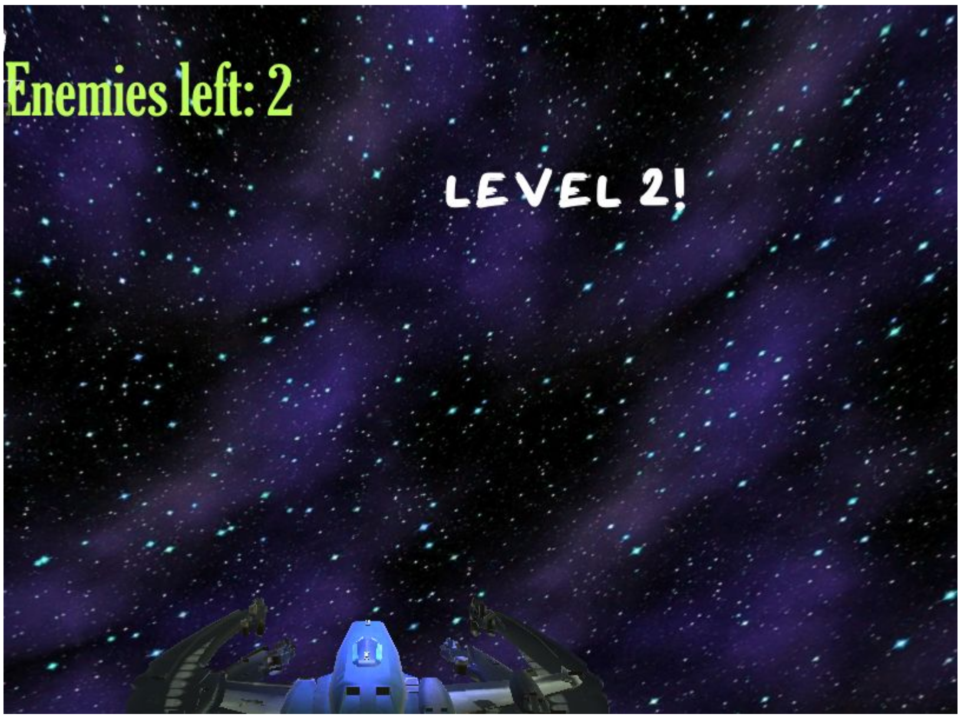
The leveling procedure we chose is quite simple but it utilizes Unity's CoRoutines (or IEnumerators) in a fun way. To display all the levels we are using Unity's Coroutines, which allows us to wait until a certain condition is satisfied (for instance, all enemies are killed) and only then proceed to the next level as shown in figure LS.0:



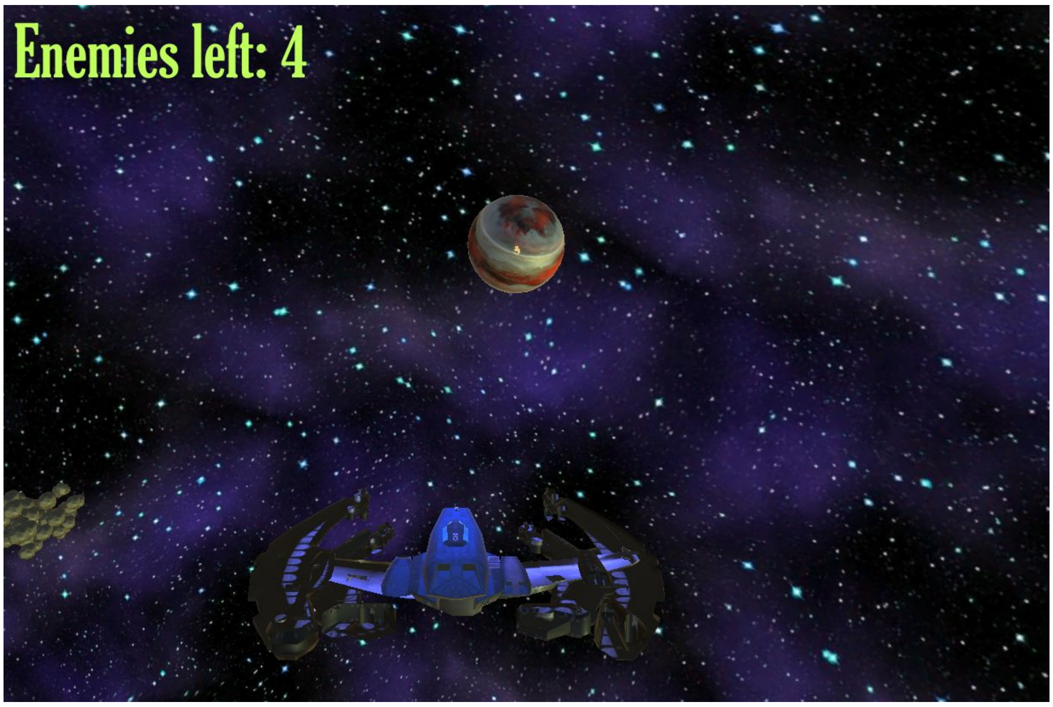
```
IEnumerator LevelSystem()
{
    if (isServer == false) throw new Utils.AttemptedUnauthorizedAccessLevelSy
    yield return new WaitUntil(() => playersInGameExist);
    while (level <= maxAllowedLevels)
    {
        shouldAdvanceLevel = false;
        SpawnPlanets();
        showLevelFor4Sec = 0f;
        StartCoroutine(SpawnLevel(level));
        yield return new WaitUntil(() => shouldAdvanceLevel == true);
        ++level;
    }
    // Win Game! => Define Behaviour!
    Win();
}
```

LS.0 - shouldAdvanceLevel is updated elsewhere, which allows the continuation of the while loop in the image. Coincidentally, "SpawnLevel()" is in itself a coroutine as well!

- From the players' perspectives, every time they advance a level a message appears for some time (approx. 4 seconds) informing them that they've leveled up.
- To be **kind** to our players, we also display the amount of enemies that are currently roaming around the galaxy, planning to attack them.
- The amount of enemies grows quadratically. For example, level 6, one should expect $6^2 = 36$.
- With every spawning enemy, the "Enemies Left" increments. (Progressing Growth) as shown in picture LS.2.



LS.1 - Transitioning from Level 1 to Level 2

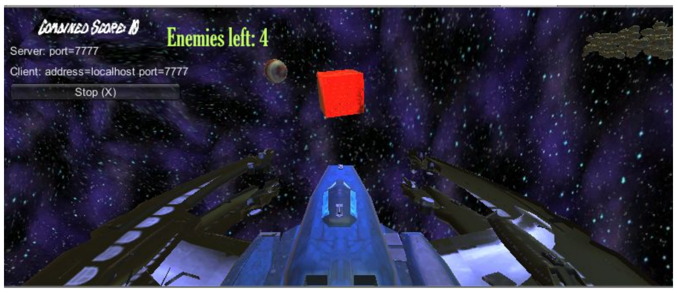


LS.2 - All enemies spawned in level 2, can see that 4 enemies are alive

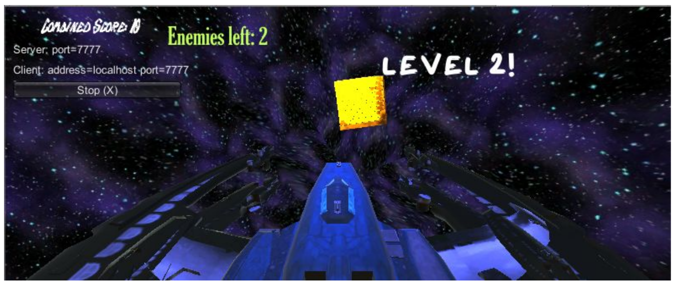
Gifts:

By killing an enemy there’s a 40% chance of a gift being spawned, player can collect the gift by flying into it.
The gift is made of a woodbox that is colored depending on which powerup the gift contains:

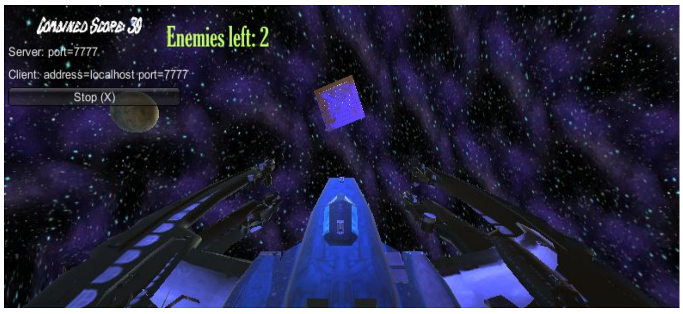
★ *Red for extra Master rocket*
Extra master rocket is available all the time until player use it, also if regular three master rockets are available, he’ll use them first after that he’ll use the extra. There can only be one extra master rocket.



★ *Yellow for extra Score*
it will add 25 points to the score



★ *Blue for extra Speed*
players speed is twice as the original.



Game effects:

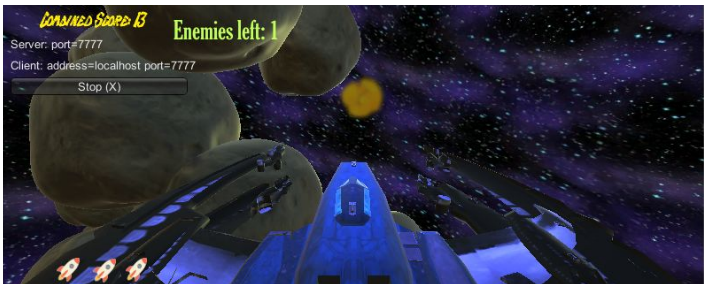
We added sound and explosion effects.

Sound effects:

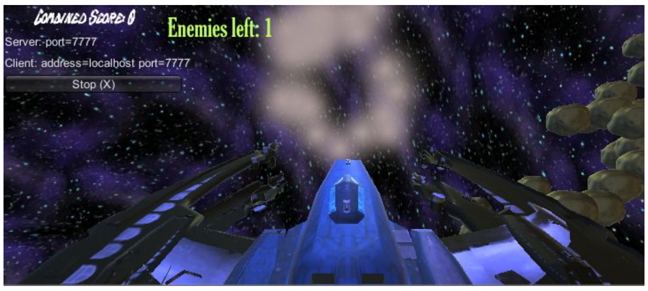
- Background music.
- Rocket shooting sound.
- Explosion sound.
- Game over sound.
- Hazard sound (when getting close to the boundary).
- Winning gifts sound.
- When master rockets are available.
- Winning the game sound effect.

Explosions:

- Regular rocket explosion:



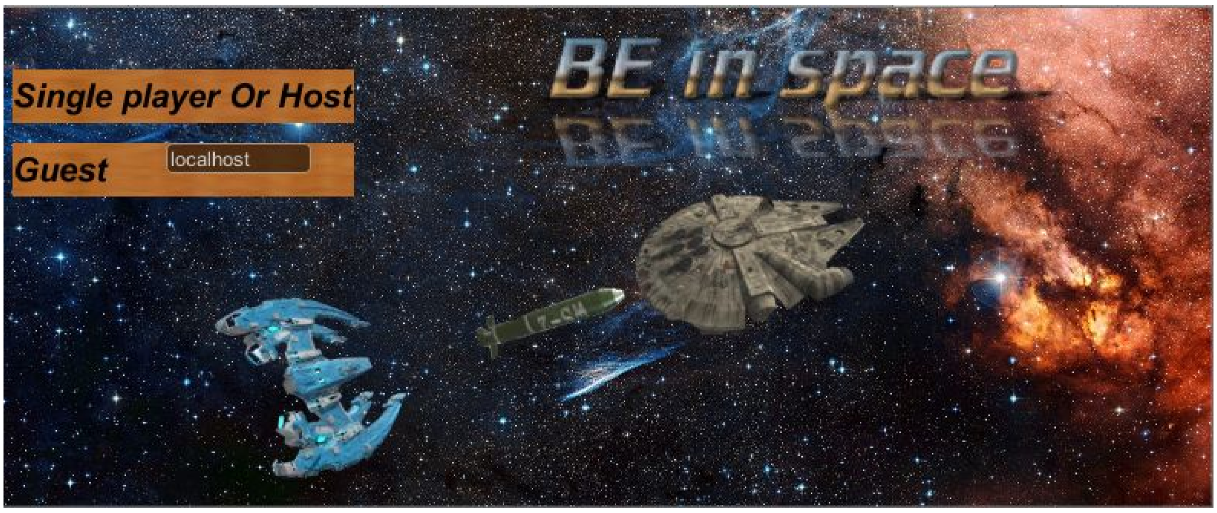
- Master rocket explosion:



Graphics User Interface:

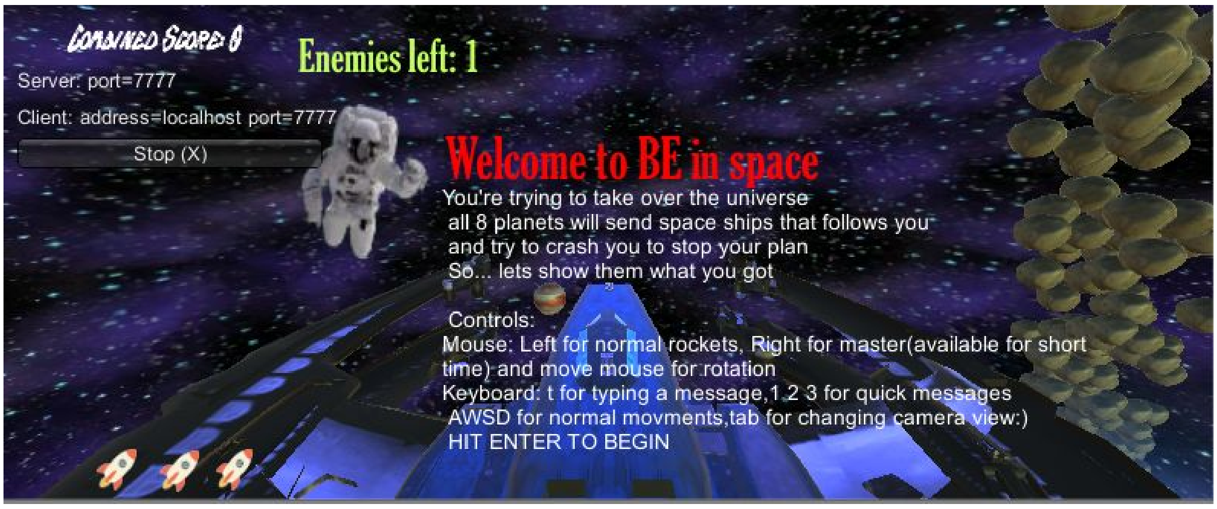
We have changed and added some friendly GUI items

Menu:



If the user chooses single player or host he'll be passed to the next scene, if Client is chosen, a host IP is required. Once provided, hit Enter then wait to connect.

Before Level 1 begins:



Welcome to BE in space
You're trying to take over the universe
all 8 planets will send space ships that follows you
and try to crash you to stop your plan
So... lets show them what you got

Controls:
Mouse: Left for normal rockets, Right for master(available for short time) and move mouse for rotation
Keyboard: t for typing a message,1 2 3 for quick messages
AWSD for normal movments,tab for changing camera view:)
HIT ENTER TO BEGIN

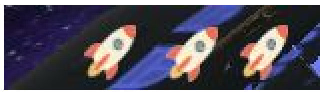
In this scene we explain to the player the game controls and game purpose. The game won't start until player hits enter.

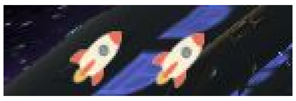
Other GUI items:

Current level:

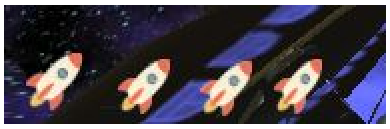
appears in the beginning of each level; shown for 4 seconds

<u>Master Rockets GUI:</u>





Player can see how many master rockets they have by looking at the number of the icons describing a master rocket, extra master rocket that player gets from gift will be added to the left of the screen.



Chat:

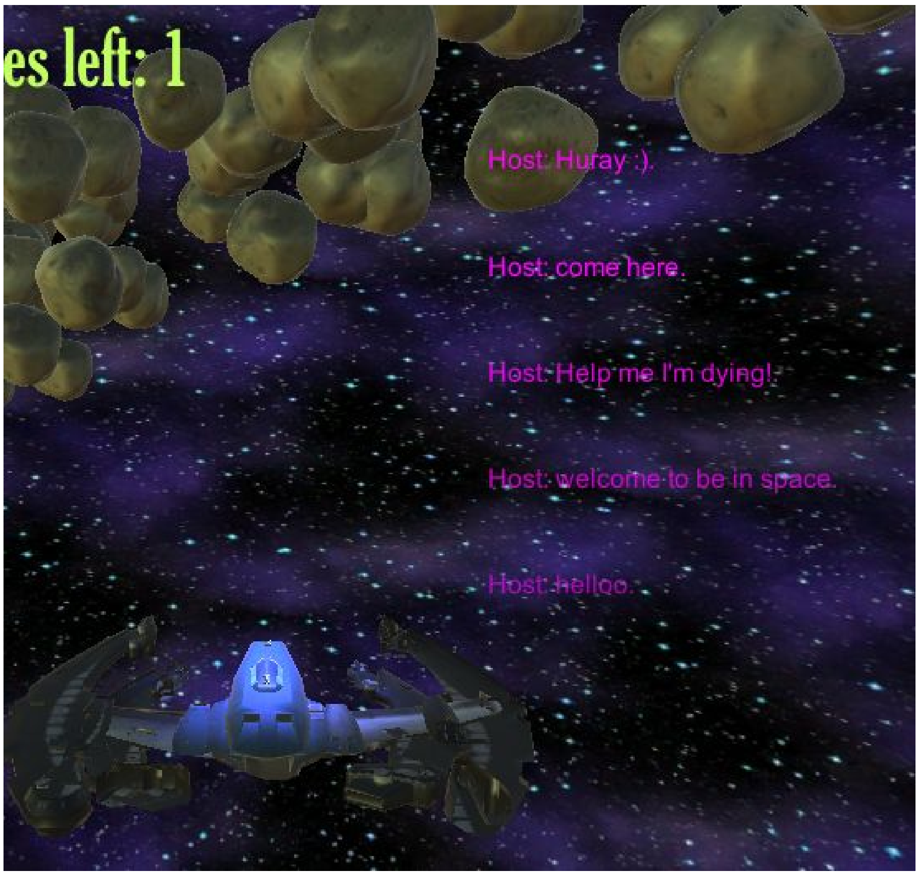
We have implemented a chat between the players so they can send each other messages. Two types of messages exist:

1) **Quick messages:**

Players can send quick message by clicking numbers 1,2 and 3
1 is for “Help me I'm dying!”.
2 is for “come here”.
3 is for “Hurray :)”.

2) Regular messages:

Players can send any message they want by hitting ‘T’ key and then start typing the message they want to send and then hit ENTER to send it, important to note that player cant move or shoot until he finished his message.



Host: Huray :).

Host: come here.

Host: Help me I'm dying!

Host: welcome to be in space.

Host: helloo.