**Design Log Group 6**

After receiving the brief from the university we had a meeting and after a brainstorming and a quick develop on everyone ideas we decided to make a futuristic cyberpunk version of “Gauntlet”. We came up that idea since the story we thought of was about an AI escaping from a network and we felt this fit well with the gameplay. The artists then started sketching so we could have an idea of what the game would look like. The designers thought of the basic story of the game and some ideas for the characters. The programmers started work on the prototypes.

We worked together to create the Project Concept Document and handed it in on time with all sections covered.

Since one of the main elements that makes Gauntlet stand out is the distinct characters, being the Warrior, the Elf, the Wizard and the Valkyrie, we wanted to make sure our game keeps that theme of player choice and allows for different styles of gameplay. We came up with these original ideas but were open to the idea of changing some if they didn’t work properly in gameplay. The enemies have different health, attacks and they each have a special power. They are similar to the original Gauntlet game but the characters have new unique powers that fit the network theme of the game.

These characters were based on the original characters from Gauntlet but with a cyberpunk theme.

Worm - Warrior

Short dash places a copy of itself down, play as the copy, old version controlled by AI and lasts for a certain amount of time

Virus - Wizard

Infect enemies with poison, nearby enemies are infected on contact

Trojan - Elf

Disguise as enemy, isn’t targeted during this time

Backdoor - Valkyrie

Player disappears and can place a portal using cursor that the other players can use.

When deciding on the controls we decided on using mouse and keyboard as a control scheme.

We brainstormed and decided we wanted to make the game procedurally generated. This would mean the game would have huge replay value and would make replaying the game going for the high score much more fun and varied. Since don’t have a lot of programming experience this was a challenge at first but we did some research and used a “digger” system to plot out a map. The remaining tiles that the digger didn’t dig were filled in with walls. This created a small room with branching paths which worked for when we first playtested the game and made sure the gameplay was fun.

Next up on our agenda was to create the game design document. This is a document which consists of all the different features which will be in the game once it is completed.

This contained many different features which we as a group decided would help our game to reach its full potential and be the best it could be. This included the following:

-An overview of the game, an analysis of the design process of both the characters and the game setting, the mechanics of the gameplay this includes; the player movement, player shooting mechanic and the different forms players can take. It also includes level designs, player goals and the assets of the game.

This document is a great tool for the entire group as it means if anyone if struggling to understand the outline of the game, they are able to look back and remind themselves!

If anyone is interested in reading it, here’s the link:

<https://drive.google.com/open?id=1nDyQjArx-An3QsTmESIGoltDSfjBTEia>

We then made prototypes.

One prototype of the game was required to highlight the basic gameplay features, the other prototype showed a more advanced feature.

The first prototype for our game was a brief demo. It contained an enemy and a player in a small area where the player could shoot the enemy. This showcased the player movement, shooting mechanic, various character options and the primary gameplay feature to defeat the enemies.

The second prototype developed by our group demonstrated our random level generation. In this prototype a new level with different rooms were generated which the player could walk through and explore. This prototype ensured that the random level generation worked in a playable state.

The two prototypes assisted us in analysing the game and ensuring that it was feasible to further develop. We initially planned to have our game controlled via a mouse and keyboard. After creating the prototypes we realised that it would be more intuitive to play the game with a controller, as it would be easier for the player to learn in a short period of time.

Here you can download the executable!

<https://drive.google.com/open?id=1WjTPbFvqG8yvOOBkTvefQQhN7O0DYaQL>

We refined some player character ideas and changed some out for ones we thought would work better in gameplay.

**Worm** - Red - High Health - Base attack: Mid range, Low damage  
**Power up**: Shoots a projectile that costs one life and if it hits, it adds the enemy as a companion (There will be a maximum amount of companions)  
**Gameplay**: Needs to build up a consistent army to be able to deal damage, but has to be careful with their aim to avoid losing too much health

**Trojan** - Green/Violet - Low health - Base attack: Long range, High damage  
**Power up**: Disguise as an enemy for a short period, that makes you undetectable to enemies.  
**Gameplay**: Shoots from distance, and has a stealth ability to reposition themself to get out of danger.

**Backdoor** - Blue - Medium health - Base attack: Short range or melee, High damage  
**Powerup**: Can perform a small blink that can teleport them through walls  
**Gameplay**: Try to progress as quickly as possible, avoiding bullets with high mobility and assassinate them closing the gap between

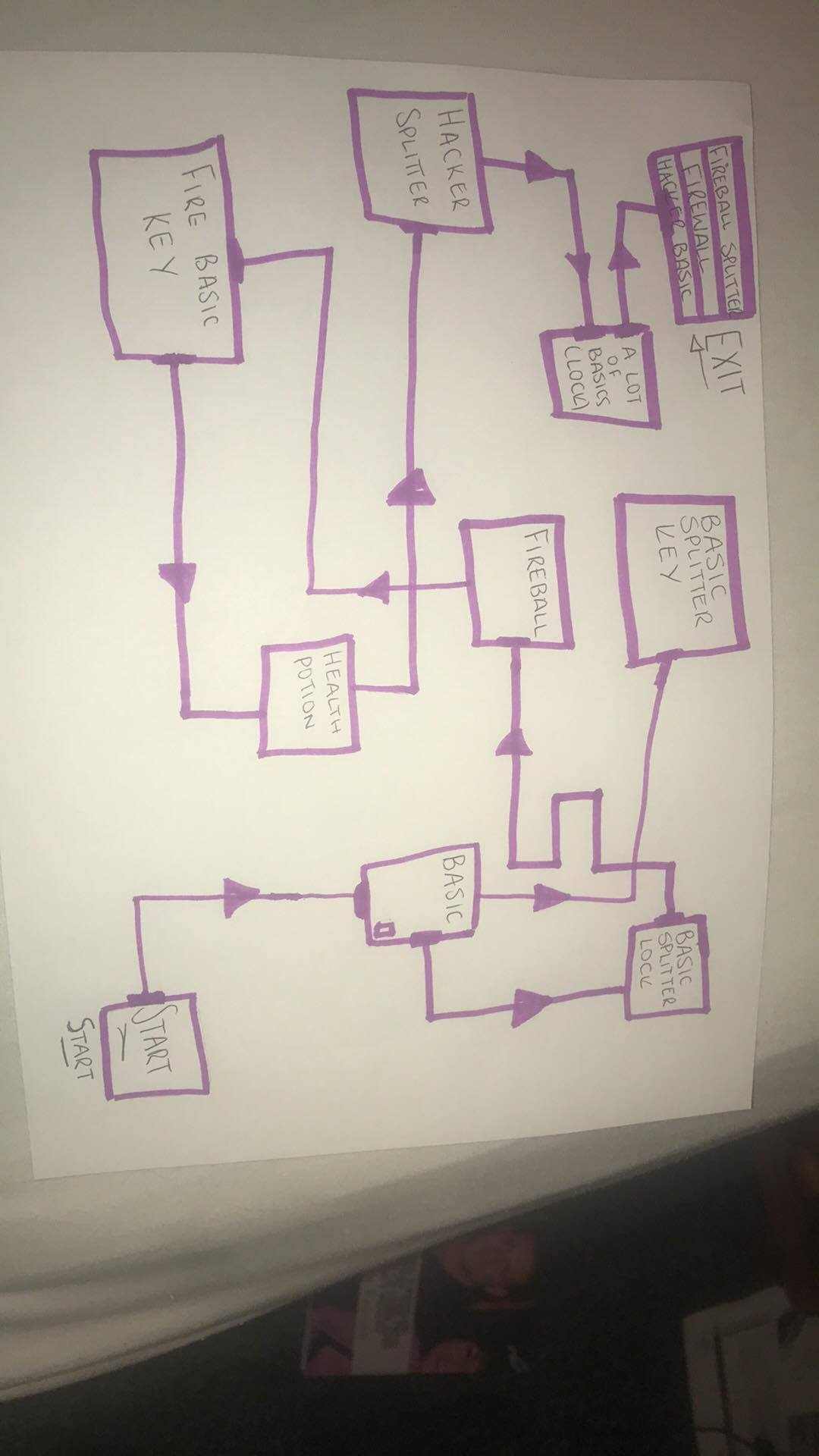
**Phisher** - Yellow - Medium health - Base attack: Mid range, Medium damage  
**Powerup**: Can place baits that will attract enemies and then explode after a certain amount of time  
**Gameplay**: Proceeds slowly using his power-up for tricky situations. Since it is really powerful, the powerup will have a long cooldown

At this point of the project, we decided to change our generation algorithm to be less linear and avoiding overlapping rooms. Since the previous algorithm was too difficult to adapt, we decided instead going for another one found on the internet while looking at some other games examples. The ones that we found permitted us to keep a procedural room locations/dimensions while keeping some pre-designed rooms that would have kept the game interesting.

We decided the main menu should fit the network theme so we designed an icosphere that looked like connected places, so that the menu would fuse with the game world. We made one that spins in the background of the main menu to create a moving network feel.

The icosphere is generated using a script and is customizable in different ways to be easily seen into the game and modified to fit the original idea.

For the interim prototype we were having issues with the procedural generation code and getting it to function properly so we decided on making a predefined level instead, this was made on paper at first and then slightly modified when creating it in Unity. This made sure that we could show off all features of the game while making sure the game would function correctly during the mini presentation.



While coming up with the level design we came up with some first concept enemy ideas.

Chaser

Follows enemies when it sees them, basic enemy type.

Splitter

Normal Enemy behaviour but when it is killed it splits into 2. 64, 32, 16, 8, 4, 2 ,1.

As they split they become smaller, faster, have lower health and do less damage.

Firewall

Big wall in the level that spawns enemies at a constant rate and has a large amount of health. Player will need to kill enemies while also attacking the firewall to get rid of it.

Fireballer

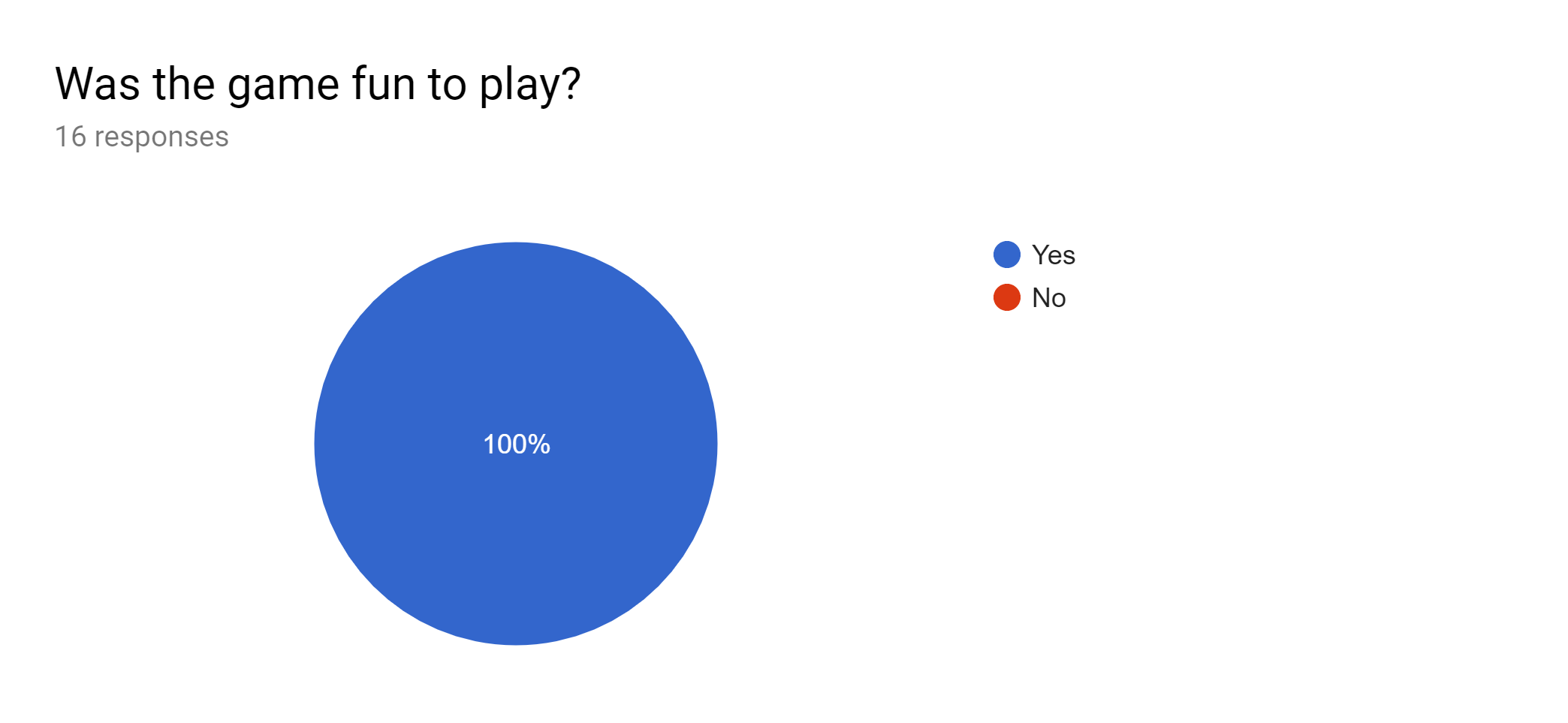
Similar to the enemy from the original game, waits for the player to get near and then shoots fireballs at them which do a lot of damage. They move slowly.

Hacker

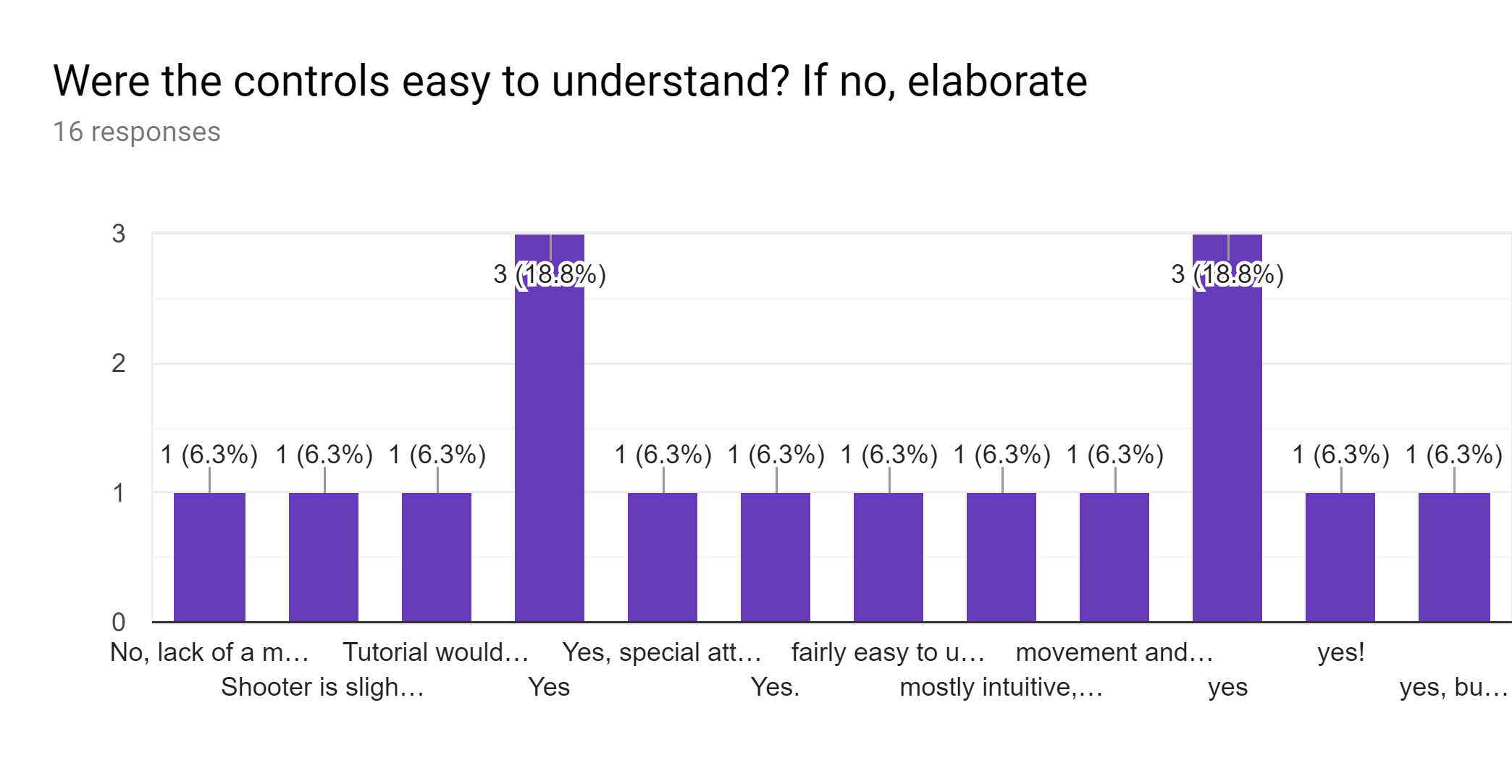
Appears in large rooms and buffs a random enemy, buff will be faster speed, more damage, more health etc

In the final stages of the game due to time and programming constraints we scrapped the idea of multiple playable characters and went with one main character, The Worm. We also didn’t manage to implement the Firewall, Fireballer or Hacker enemy types.

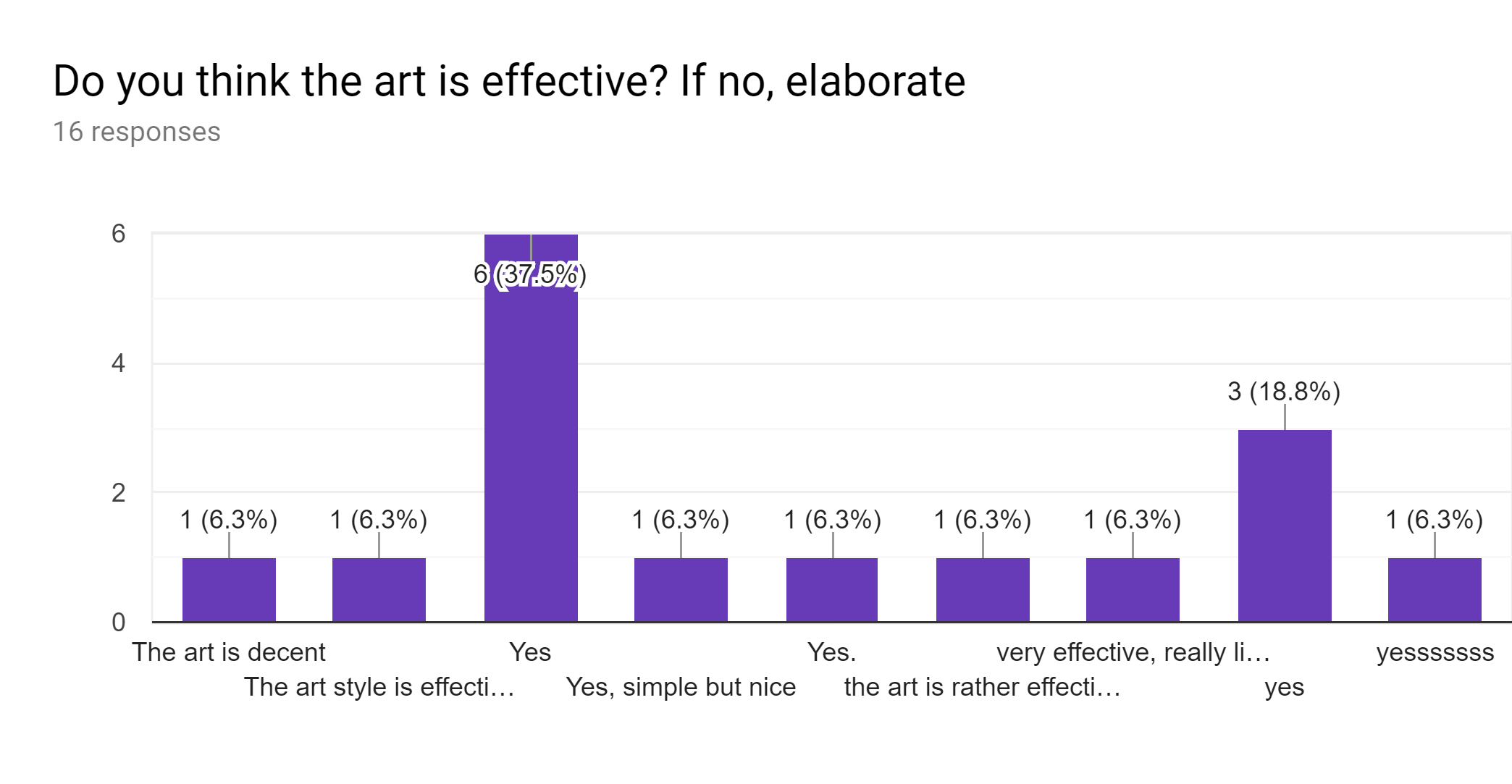
Playtest Report



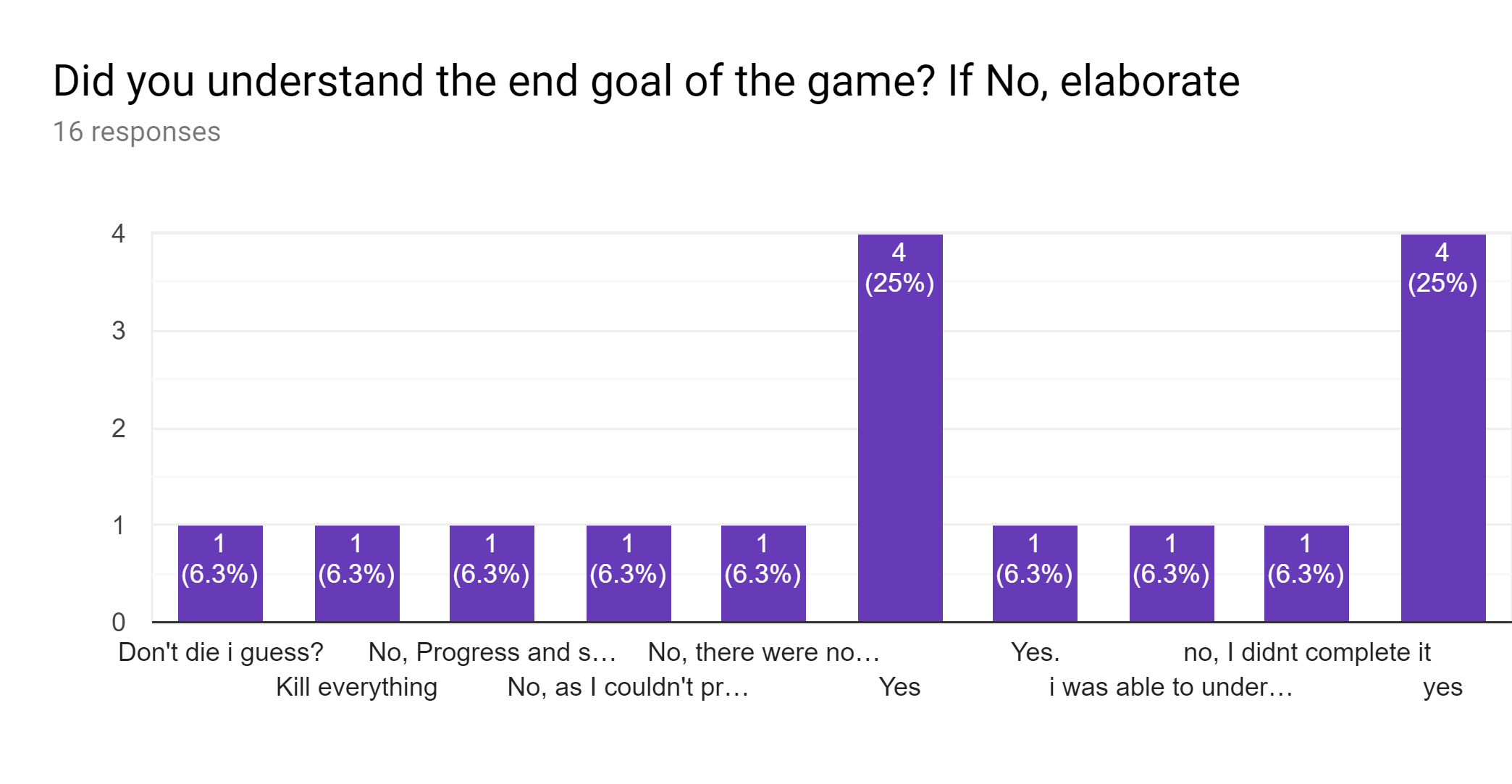
Everyone thought the game was fun to play.



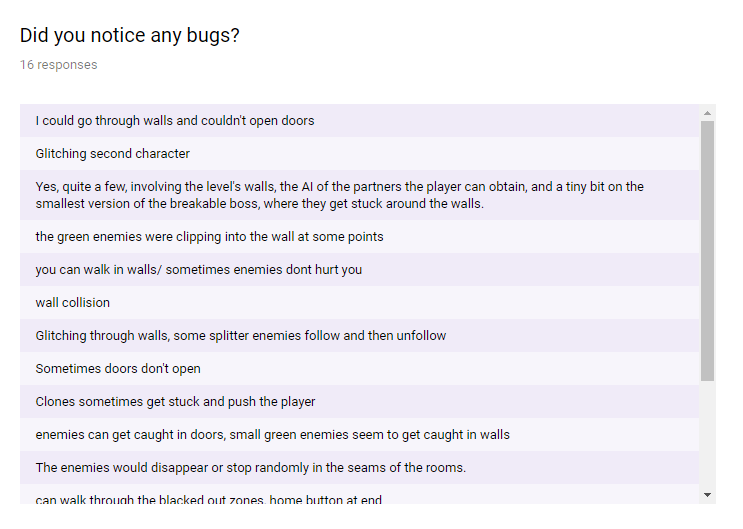
Most people thought they were but some people said a “how to play” section would help. This was created as an image but was not implemented into the game due to time constraints.

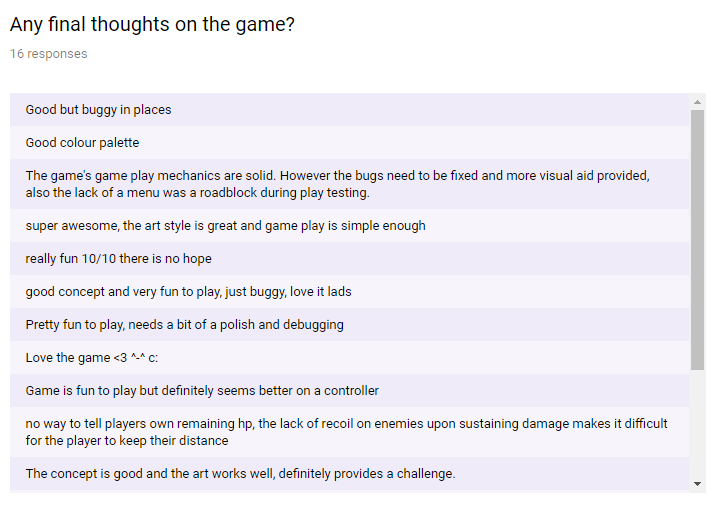


Everyone thought the art was effective, “players and enemies stand out”



Most people understood the end goal but some were confused as to what was going on. We thought teleportation through the rooms instead of using corridors made sure that the player always knew what direction to go in. Also using cutscenes for the story would give purpose to the dungeon, as the story states why you are trying to reach the exit.



Here are some bugs playtesters found. We managed to fix a few of these but could not fix the others due to programming constraints. We fixed clone movement, some enemy code that was not working, the aim system, the player special ability and a problem with rooms that were cleared but didn’t unlock doors.

We observed from the tests that from a gameplay point of view the game is fun to play.