**COMPSCI 113**

**Computer Game Development**

**Game Design Document**

**Super Space Blasters**

Team Night

Team Members:

* Stephanie French
* David Hess
* Christian Sydow
* Benjamin Loh
* Alex Dunn

Description of game play, key features, win conditions:

* 2D platformer
* Multiplayer arena deathmatch
* Up to 4 players
* Short rounds (players can only take a few shots before dying)
* Win decided by user-determined condition (first to ten, best of five, etc.)

Genre and important influences:

* Multiplayer action game
* Heavily influenced by Duck Game (<http://store.steampowered.com/app/312530/>)

Background story:

* Aliens prisoners are forced to go to random planets and fight each other in the Galactic Gladiator tournament.

Art design: Using pre-made art

* So far using assets from <http://kenney.nl/assets>
* May get others if we find it necessary along the way (will properly source)

Technical platform (Libraries, target operating system/device):

* Engine: Unity
* Programming Language: C#
* Target OS: PCs (whichever operating systems are supported by Unity)

**Game Specifications**

**Player Interactions:**

* Start game
* Play round
* End round
* Win game

**Type of Game:**

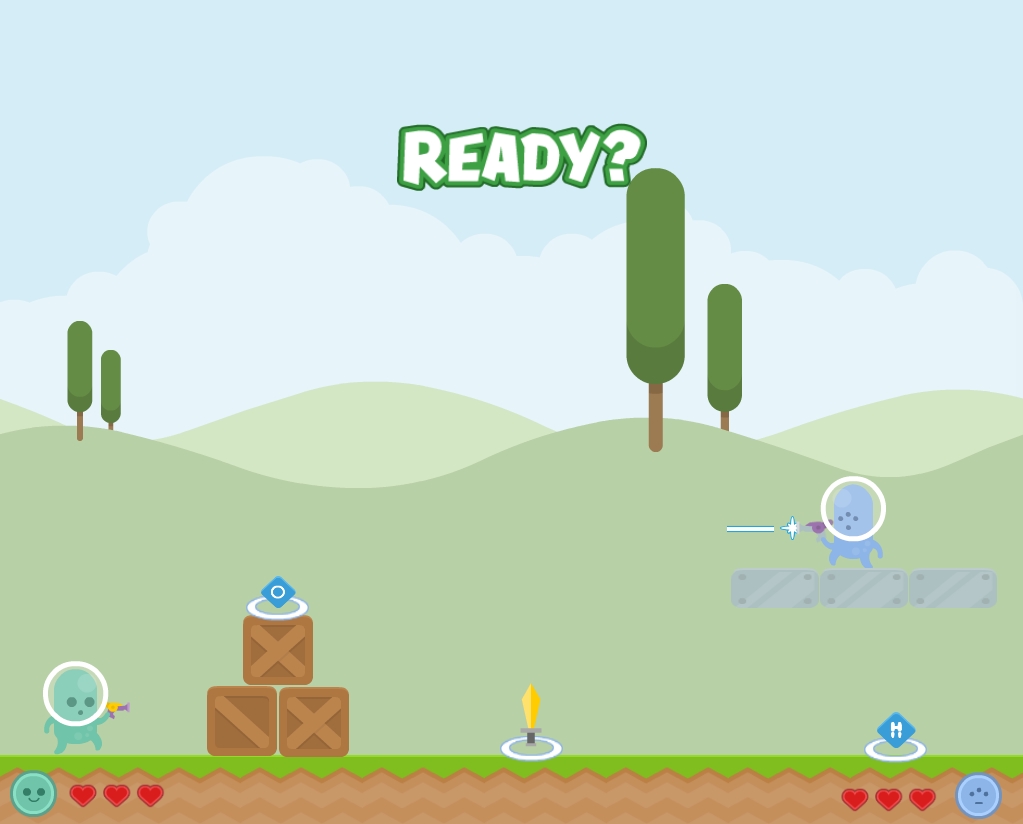
* 2D multiplayer arena platformer
* Overall gameplay is a multiplayer battle arena focused on fast-paced combat

**Rules/Mechanics**

* Multiple game modes which are selected before the start of the game
* Played in **matches**, which can consist of multiple rounds depending on the game mode
* Players who win the match are awarded currency
  + 1st place gets 20 coins
  + 2nd - 10
  + 3rd - 5
  + 4th - 1
  + Used to buy weapons that are used in the upcoming match
  + Currency is not saved after turning off the game, and only persists in each individual instance
* **Best of X rounds:**
  + At the beginning of the match players will agree upon a set number of rounds
  + Each round consists of a single elimination deathmatch between all the players.
  + Last man standing wins the round
  + If everyone dies then the round results in a draw and no points are awarded
  + After each round, the arena is changed
  + Once the set number of rounds have been played out, the player with the most victories is the winner.
  + If there is a tie between multiple players in number of rounds won, there is a sudden death tie-breaker round
  + The winner of that round is the final winner
  + Ties are prevented in the tiebreaker by ending the round as soon as the next to last player dies
* **Deathmatch:**
  + This gamemode consists of a single round where players are given at least one life.
  + Players will agree upon a set number of lives.
  + When a player dies if they still have leftover lives they will respawn after a short time.
  + When a player runs out of lives they are eliminated from the match
  + When all players but one are out of lives, the player with at least one remaining life is the winner

**Arena Layout:**

* Each round will take place in a rectangular arena
* Arena is a two dimensional grid of “blocks” that create a traversable environment.
* Include things like:
  + Platforms
  + Slippery ground.



**Movement:**

* Players can move up, down, left and right in a two dimensional arena.
* The arena will consist of platforms which is generally rectangular in shape.
* Collision between players and environment will be that players can stand on top of and cannot travel through platforms.
* Possibly create player on player collision. Undetermined
* Movement will be generally fast enough so that the entire arena can be traversed in 5 to 10 seconds
* Up and down movement consists of jumping and falling respectively
* Jumping will roughly cover a distance of 2 “blocks”

**Artwork and user interface.**

show screen shots, sketches, prototypes, samples.

Describe the overall "look and feel" of the game.

**Gameplay and balance.**

What can the player(s) do?

* Jump
* Move
* Pick up weapons
* Pick up items
* Use weapons
* Use items

What can the players see and hear?

* The game map and all players will be on screen at all time
  + Depending on the size of the maps, we may implement some dynamic zoom to stay as zoomed in as possible while making sure all players are visible.
* Status, health, points of players will be visible in various positions on the screen
* Background music will be playing (can be disabled under options)
* Sound effects from player movements, weapons, power-ups, etc. will be audible

How does he or she interact with the game world?

* Basic platforming concepts (running, jumping, swimming?)
* Using weapons and power-ups to affect the player and opponents in various ways

How are weapons or other tools acquired, used, lost?

* Scattered around the map in predetermined spots
* Weapon spawners will generate a random weapon from a preset pool of weapons. On a timer. When there is a weapon already a weapon at the weapon spawner, no more weapons will be created.
* Players run over the weapon to equip it. Already equipped weapons are discarded and destroyed.

What kind of physics model, if any, is being used?

* Unity’s own physics model will be used.
* Players can collide with one another
* Slanted ground which causes players to fall towards the incline of the slope

**Weapons:**

* All of the players spawn with the basic weapon (ray gun) at the start of the match. They would pick up new weapons scattered at various spawn point positions on the map.
* Sword:
  + Basic Weapon
  + Medium swing speed
  + 3 Damage
  + Destroys projectiles on hit
  + Melee range
* Ray Gun:
  + Basic Weapon
  + Medium firing speed
  + 1 Damage
  + Fires a fast straight moving laser projectile that travels a short distance
* Super Ray Gun:
  + Upgraded Ray Gun Weapon
  + Medium firing speed
  + 1 Damage
  + Fires a fast straight moving laser projectile with unlimited range.
* Force Gun:
  + Very Fast firing speed (several projectiles/sec)
  + 0 Damage
  + Projectiles travel very fast and push anything they collide with
* Shotgun:
  + Slow firing speed
  + 1 Damage
  + Fires three projectiles in a cone shape
* Bazooka
  + Slow firing speed, easily dodgeable
  + Strongest weapon
  + Single Shot
  + 3 Damage
  + Fires a slow speed projectile

**Music and sound.**

Describe the music and sound you plan to have,

* Start screen music
* Round music
* Each weapon will have a unique sound effect
* Different stages/levels will have their own music

Describe where the music and sound will come from

* Music and sounds will be made using FL Studio.
* Music will be similar to electronic and space genre to fit in with the theme of Aliens
* Premade sounds

List important sound effects.

* Player dies
* Item/weapon pickups
* Item/weapon usage
* Start/end of round

[**Intro Song**](https://drive.google.com/open?id=0B_U4m0L3ZbrEUllNZENfN0czbXc)by Benjamin Loh (<--click the link)

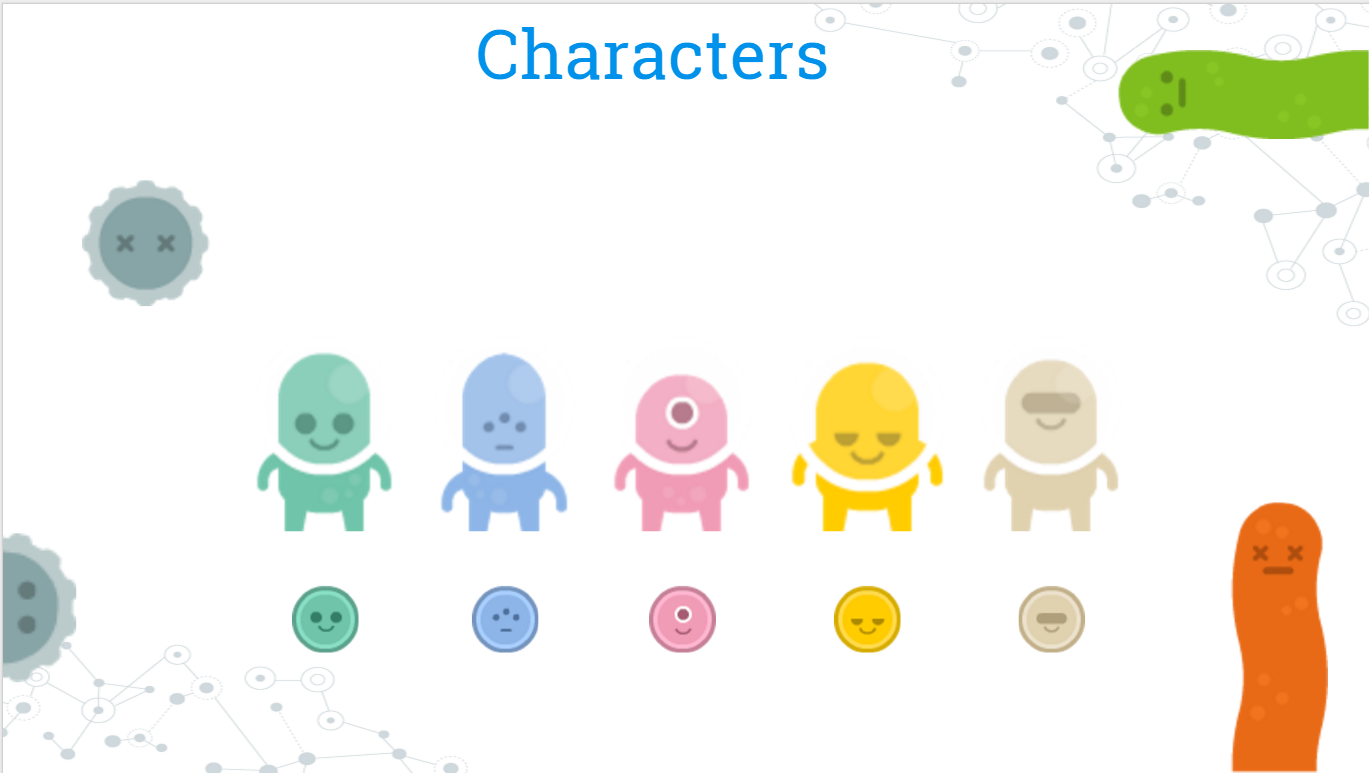
**Background story.**

Multiplayer story:

* In a galaxy pretty close, there are alien prisoners who are chosen to fight as galactic gladiators purely for the entertainment.
* Galactic Gladiators is the most popular show in the Galaxy because everyone loves watching convicts sent to die for the sake of entertainment.
* They enter the tournament for a chance to be set free from their prison sentence if they win the tournament.
* For all the other gladiators that lose the tournament, they face imminent death.

**Characters.**

* The players will choose from one of several different cartoon alien sprites. Currently there are five different colors but we may add more (although only four is necessary to have each one unique during gameplay)
* Various enemy creatures may be present on the maps as another type of platforming obstacle for the players. They will have very basic behaviors.
* Brown: Press space to activate shield: Creates front shield that blocks projectiles for short duration
* Green: Double Jump
* Pink: Press space to heal: Heals one heart
* Blue: Runs 20% Faster
  + press space to dash: Teleport forward a short distance
* Yellow: Maximum of 4 hearts (3 is normal)
  + press space to activate juggernaut mode: Enemies you collide with lose two hearts
* (Possible future addition) Bosses at the end of single player levels



**Levels.**

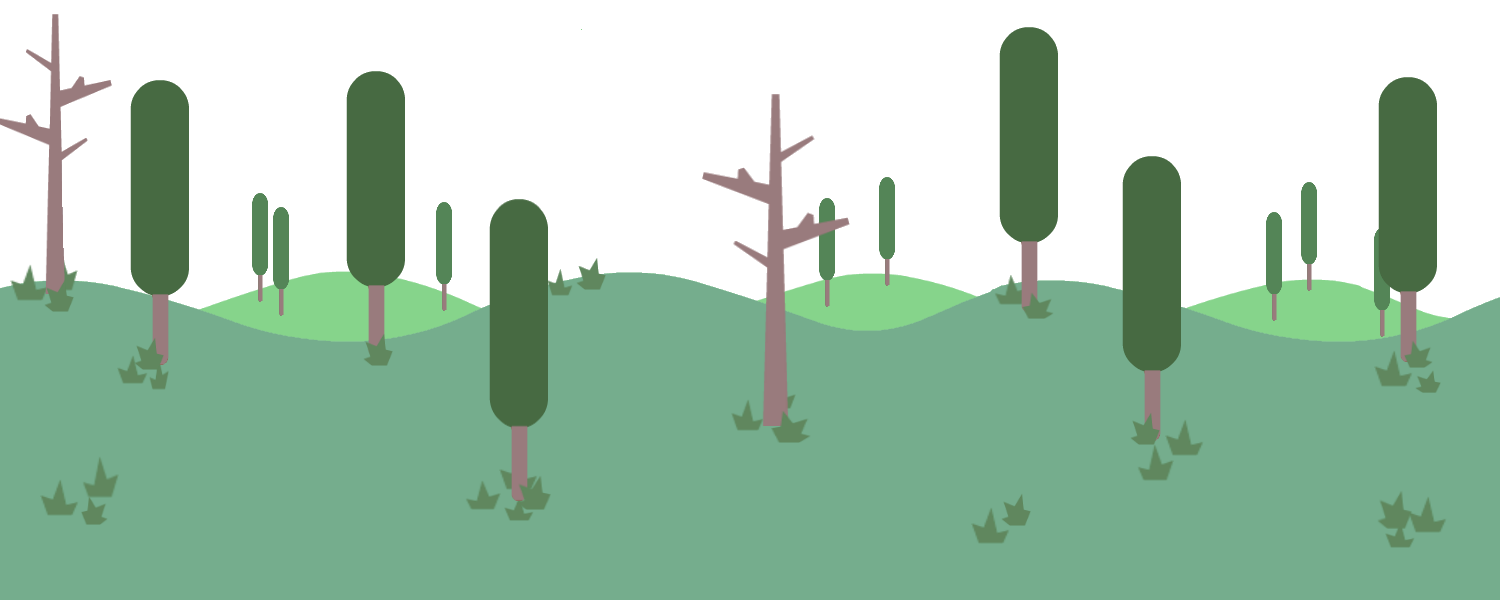
* Rounds, last player standing wins the round, ending the match
* Players can select best of (x) rounds or first to (x) wins before starting the game
* Players can pre-select levels or have random level iteration
* Levels will vary from icy, to fiery, and even candy-themes
* Icy levels will feature slippery environments which can result in players slipping off the edge or into other hazards, some weapons have recoil which can also lead to players slipping
* Fiery levels will have lava hazards and fireballs throughout the level, lava levels can rise leading players to seek shelter at higher levels of the map

Some examples of created backgrounds

Dark blue space, will be used as the furthest back layer on some levels



Green forested hills background, will be used in front of the sky layer, but behind the player



**Artificial Intelligence.**

How do NPC's, enemies, and followers behave?

* + Currently there will not be A.I.’s implemented in this game and the focus will be on player vs player interactions
  + We may choose to implement simple A.I. in the form of enemies which function more like environmental hazards. Something like a blob that moves back and forward or in a simple set pattern.

**Technical Specs**

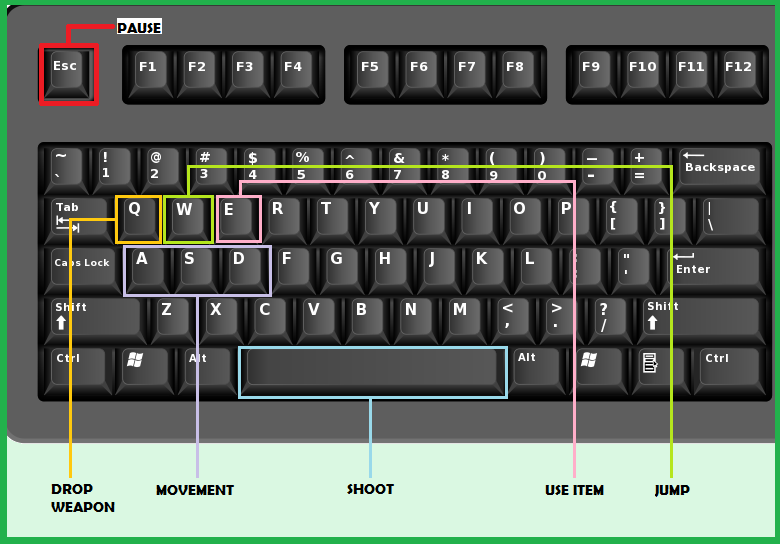
* Programming languages, including compiler and emulator.

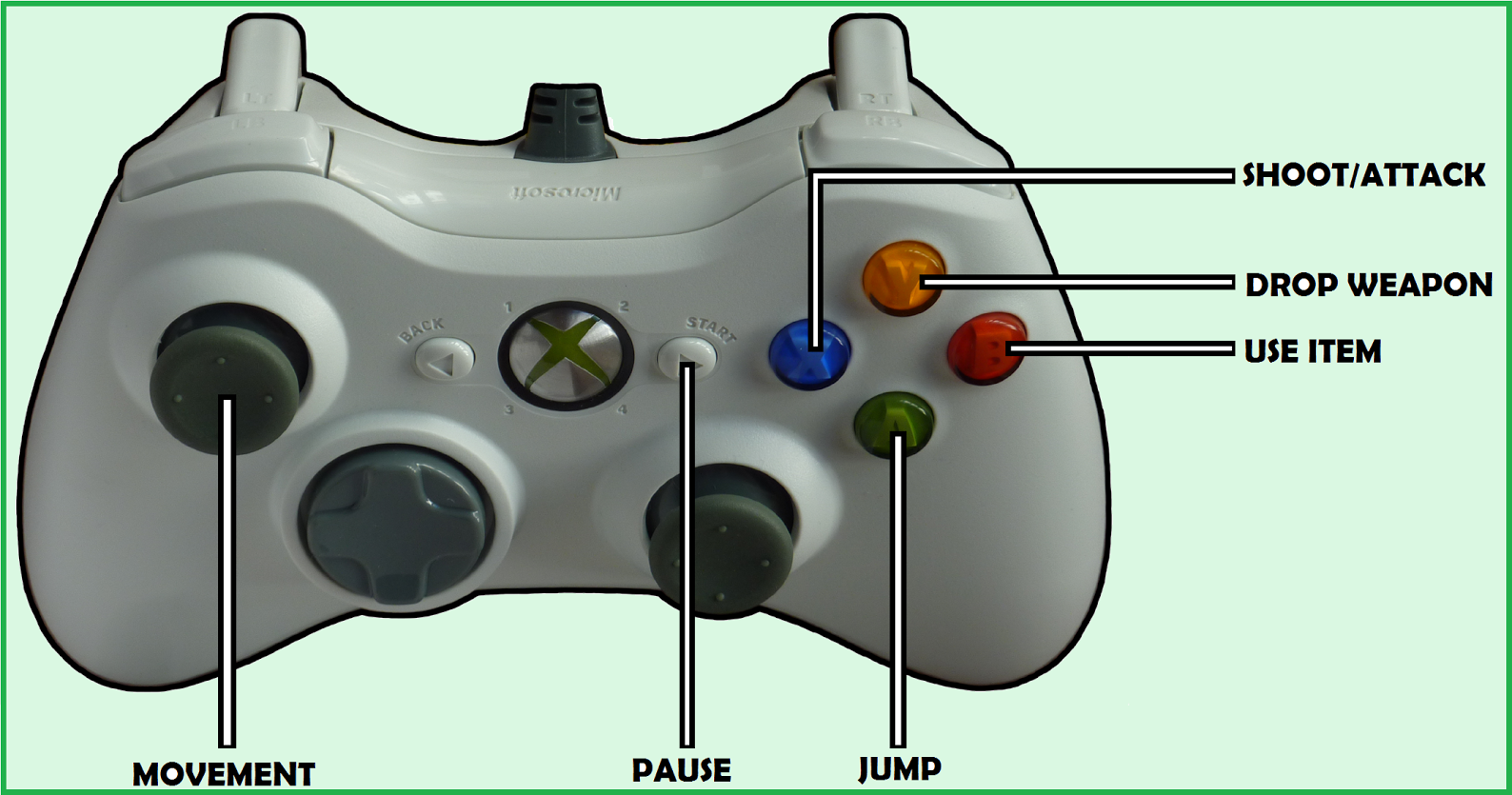
C#

Compile using Unity

* Libraries that will be used.
  + Unity Engine & Library
* Implement maps by creating a two dimensional array and fill in elements with various terrain elements
  + possibly create a random map generator
  + For randomized maps we can utilize an algorithm that forbids enclosed terrain areas. I.e. make sure all areas are accessible
* Any code or engines to be used.
  + Unity physics engine
* If you plan to use a game engine, describe it and explain what its interface (to the rest of the system) is.
* Try to implement network connectivity.
  + Ideally try to have one player per computer.
* Target hardware and Operating System.
  + Keyboard and/or mouse integration
  + Xbox controller integration

**Keyboard Controls**



**XBox Controller Controls**

**Data Structures**

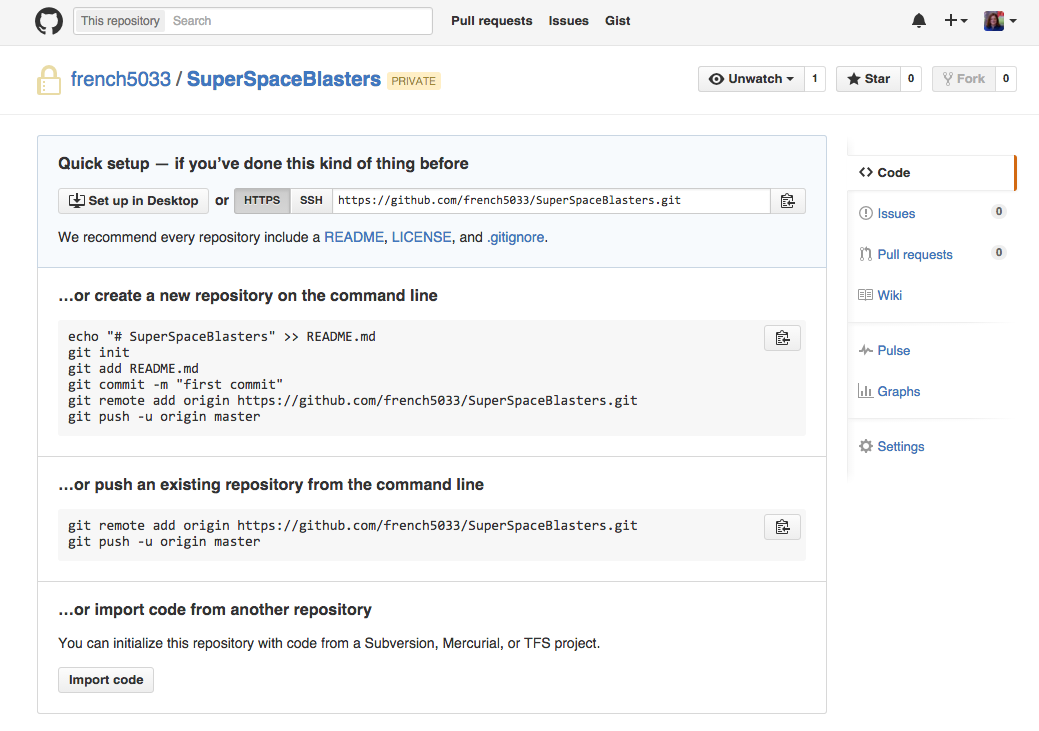
* Map
  + Name
  + Elements (below)
  + Background texture
* Map element
  + Texture
  + Dimensions
  + Coordinates
  + Type
    - Standard platform
    - Special platform (lava, ice, etc.)
    - Weapon spawn point
    - Item spawn point
    - Player spawn point
    - Destructible
    - Button
    - Other?
  + Interactivity
    - Solid platform
    - One-way platform
    - Liquid
    - Spawns weapon or item (like a Mario item block)
    - Other?
  + Functionality (if interactive)
* Player
  + Name
  + Color
  + Status
    - Alive
    - Dead
    - Other?
  + Health
  + Current weapon
  + Current item
* Weapon
  + Name
  + Type
  + Damage
  + Maximum ammo (if applicable)
  + Remaining ammo (if applicable)
* Item
  + Name
  + Type
  + Functionality
  + Maximum uses (if applicable)
  + Remaining uses (if applicable)
* Unlock
  + Name
  + Type
    - Weapon
    - Item
    - Other?
  + Conditions
    - Achievement
    - Other?
* Achievement
  + Name
  + Type
  + Conditions
    - X kills
    - X wins
    - Other?

**Algorithms**

* Damage and life are managed by a “damage handler” script in c#
  + Objects are given the script component
  + The script has a public int variable that assigns the carrier HP
  + Damage is assigned when two objects collide using the function “onTriggerEnter2D”
    - Players are not triggers therefore they do not do damage when they run into each other
    - Projectiles and powerups set off the function, but we use a condition that prevents powerups from doing damage to players
    - Currently all collisions do 1 damage but we might change that to be based on the type of weapon
  + When an object’s HP reaches 0 it is destroyed
  + Powerups/projectiles have a health of one so they are destroyed when they collide with something
* Weapon pickup: Weapons are created as prefabs in Unity. When a player collides with the on ground weapon the game object is destroyed and the corresponding weapon is created as a child of the character.
  + - A transform of the weapon is attached to the hand of the player
* Weapon firing: When a player presses the fire key a bullet is instantiated at the tip of their gun.
  + Weapons can either be fired by holding down the fire button or tapped.
  + Projectiles will be fired at less than or equal to the set firing rate of the gun.
  + The bullet has a movement script that moves it forward at a preset speed
  + The bullet is destroyed either when it collides with a target or after a set amount of time
  + Physics is handled by Unity’s built in physics engine (Rigid Body 2D)
  + Collision is handled with Unity’s built in collision detection system (box collider 2D)
  + Movement: When a player holds down the move left/right key their character moves in that direction at a set speed.
    - Players move at a constant speed and there is no acceleration/deceleration
* Jumping: When a player presses the jump key a set force is applied on their character in an upwards direction.
  + Only single jumping is allowed
    - Movement script has two booleans: land and jump
    - Jumping is only possible when jump is false and land is true
    - When a character jumps the jump bool is set to true and the land to false
    - When the character collides with an object tagged as ground the land boolean is set to true and the jump to false
* Parallax scrolling utilizing layers, different layers will have different speeds for scrolling. This will be done by creating a script that allows us to set this “speed” for different layers. The furthest back layer may move slower, and the nearest back layer will move faster, this will cause the front most background layer to move more than the back one, giving the illusion of depth
* Infinite background generation, when the camera gets to the edge of a background layer generate another (possibly mirrored to give the illusion of different tiles) background asset connected to the previous one, this will be done by copy and pasting the existing asset on the detected empty location that is upcoming, thus the script will need to check ahead to see if an empty location is almost in view of the camera

**Back-up and Version Control Plans**

Using GitHub for version control



**Schedule and Personnel**

|  |  |  |
| --- | --- | --- |
| **Week** | **Benjamin Loh** | **Alex Dunn** |
| **Tasks** | **Programmer/Sounds** | **Programmer/Art** |
| 6 | Work on the Music for the Intro/Start menu screen.  Think of themes for the different levels. | Implement collision, weapon, item pickup mechanics.  Work on art. Photoshopping bullet images etc. |
| 7 | Work on the Music for the various different maps such as the Candy level, Ice level, Fire level.  Also create sound effects for the Ray gun, gatling gun and plasma rifle. | Implement title screen and buttons.  Work on creating more weapons.  Implement damage handler script |
| 8 | Work on the sound effects for the remaining weapons such as sword, laser sword, missile launcher and squirt gun.  Create the sound effects for the powerups such as jetpack, double damage, invisibility and juggernaut. | Work on game mechanics (i.e. who wins the round, starting new rounds, hero select) |
| 9 | Make the sound effects for the traps like pitfall, levers and doors.  Implement and program the music and sound effects to the game.  Help to code remaining things needed for program. | Implement screens for level select and character select.  Continue working on guns/powerups  Making sure the multiplayer is working as intended. |
| 10 | Work on the program to finish up any loose ends.  Test out the game to see if there are any characters that are overpowered and need to be nerfed. Try to make characters as balanced as possible  Try to find potential bugs in the game. | Work on game polish.   * Animations/graphics look smooth * balance is reasonable * movement feels fast/fluid * Bug checking |

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Stephanie French** | **David Hess** | **Christian Sydow** |
| **Tasks** | **Programmer, Design** | **Programmer, Networking** | **Programmer, Design** |
| 6 | Program characters/abiliites | Researching networking methods/libraries | Create different weapon sprites (Missile launcher, ray gun, sword, laser sword, squirt gun, etc.)  Create backgrounds for levels (Fire level, Ice level, forest level, space level, etc) |
| 7 | Check that attacks, collision work  Add different attacks  Work on desert scene | Researching networking methods/libraries | Finish any sprites and backgrounds not yet done  Add parallax script to game  Maybe add infinite background/foreground generation (might not be necessary) |
| 8 | Add characters & scenes  Check for any functionality errors  Display health on scene | Implementing networking  Working on multiplayer (local/network) | Create levels with blocks for actual level (platforms, hazards etc)  Create blocks for levels if needed (Different shapes, new hazards like fire or lava) |
| 9 | Finish up any remaining design/abilities for all characters | Implementing networking  Working on multiplayer (local/network) | Finish up any remaining design for levels, adding weapon spawn points (possible random weapons) buff spawn points, player spawn points |
| 10 | Finish up coding on any extra features for the game:  Create additional scripts for any remaining power ups that need to be implemented (Jet Pack, Invulnerability, etc.)  Create AI for level monsters (fire worm, slime, saws) | Implementing networking  Working on multiplayer (local/network) | Add vertical parallaxing to levels, create menu and title screen interface. Add sprites for power up effects (bubble for invulnerability, jet pack for jet pack) Add any additional artwork to levels, or if time implement more levels (Space level, desert level, water level) Create randomization spawn of weapons and power ups on level locations |