Backyard Snowdown

Game Concept

Backyard Snowdown is a top down dodgeball like game, where players fight in a snowy backyard with the added flair of a child's imagination. Players can pick up and shoot tennis balls, or use a plethora of abilities such as snowballs or snowmen to take down their friends.

Features

Shooting/Throwing (Projectiles): 2 projectiles on the field that players must fight over and then shoot/throw at other players.

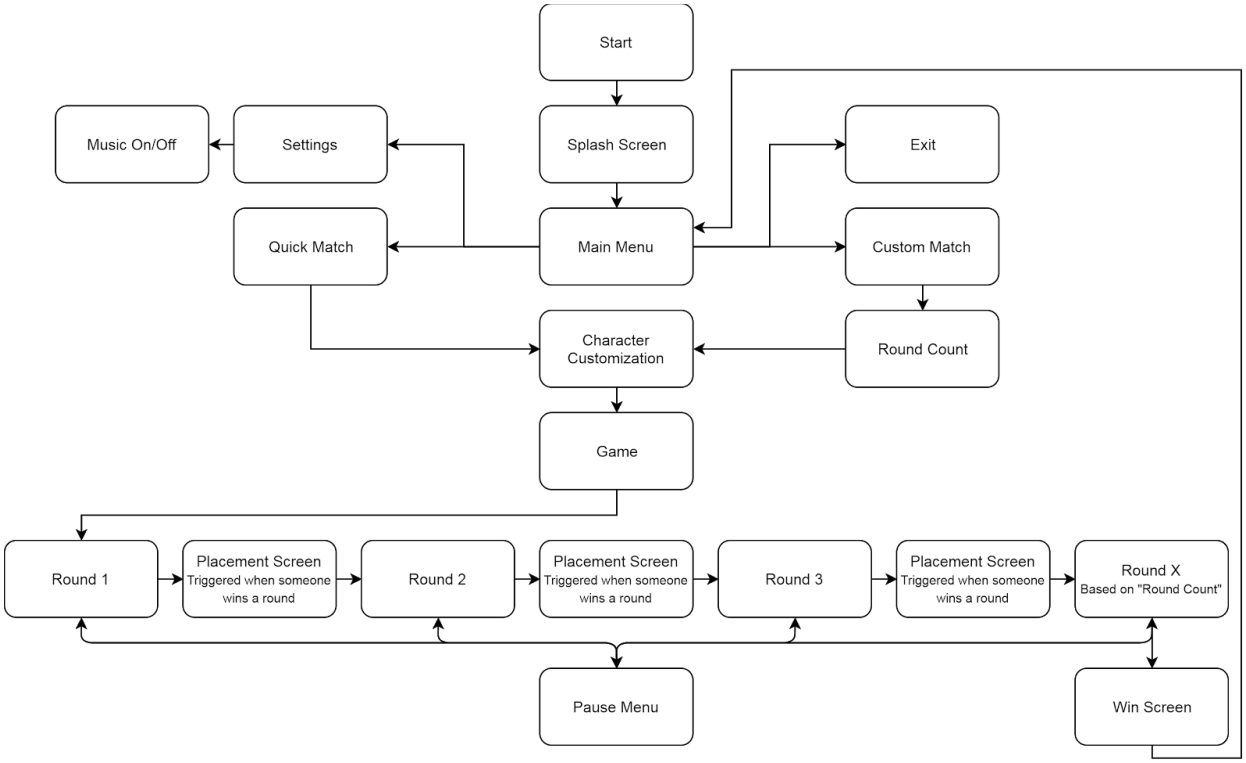
Movement: Players can be controlled via the analog sticks, multidirectional movement right stick to aim.

Dashing: Players can dash to dodge projectiles.

Technical Risks

- Unity - Unity is still a bit new to the programmers so some risks are just that some parts of the gameplay that they will need to research how to program/implement.
- Controller support - I have been told that Unity can struggle a bit when using multiple controllers on one PC. This means that a risk could be just ensuring that there aren't any horrible glitches that could be detrimental to gameplay due to controllers.
- Navigating menu with controller. Using the cursor's position is relatively simple but using a controller to change selected options may be challenging.
- Balancing the game/abilities so that all of them are equally viable. If there is one that is clearly over-powered then there isn't much point in making the others.
- Charging the shoot and making the display of the charge being in sync with each other.

Game Flow



GameObjects, Scripts and Systems

Logical Objects

- Tennis Ball
    - Description: The tennis ball is a projectile that can be picked up and shot out by the player's guns. It is always in play and bounces off the edges of the map. Shooting l it and hitting someone is the main aim of the game. When you hit someone with it, they lose some health.
    - Script: As this is the main mechanic to the game, this will have a rigidbody and a sphere collider on it. Sphere colliders are more costly than a box collider but I believe that as the ball needs to roll, using physics to roll would be the easiest way to implement it. The collider will have a trigger on it that when it collides with a player, it will be parented to the player and take on its local transform until shot. Turn "is kinematic" to true so that it won't move or use physics while the player has it on them. Public variables include; Mass, Bounce count, Velocity
- Arena Obstacles
    - Description: Arena obstacles are there to add a bit of difficulty and randomisation to the gameplay. There will be preset obstacles that are common objects in a kid's backyard that the Tennis Ball will bounce off. There will be a set area that the obstacles can spawn in and it will pseudo-randomly spawn somewhere inside the allocated area.

    - Script: These are just basic objects that will have a capsule collider on them. There will be an empty gameObject that will have a script to create them on the load of the game. It will use srand. Srand uses random but creates a new seed dependent on the computer's timer.
- Static Perspective Camera
    - Description: The Static Perspective Camera is the main and only camera in the scene. It will be about 35° off top down so that it shows the nice 3D models that our artists are making. This will have an audio listener to hear the sound effects and music.
    - Script: The Main Camera will be static with no code on it.
- Snowballs
    - Description: Snowballs are a secondary ability that you can choose to use that on collision slows the target. These will be relatively basic, small, white spheres that on destroy will have a small white particle effect for a small snowy touch.
    - Script: The Snowball will have a box collider on it because it doesn't touch the ground and won't need sphere-like physics. It will have a trigger that will destroy the object, slow the target and a small AOE around them. It will also replace itself with a small particle effect. Public variables include; Velocity, Slow amount, area of effect.
- Snowman
    - Description: Snowmen are another ability that will summon a snowman in front of you as cover from incoming projectiles. It will be roughly the same height as the kids and will be in a cartoon-style with a carrot nose.
    - Script: The Snowman will have a hit counter on itself to destroy after 2 hits.
- Pogo Stick
    - Description: The Pogo Stick is the feature to an ability where you bounce on it into the air and slam it on the ground, creating an AOE(area of effect) knockback. It will be very basic as it is just a pole with handles and foot stands. It also is only used for a short amount of time then disappears, unlike the snowman that will sit there for awhile, visible to players for much longer. Keeping the pogo stick minimal polys would be good for optimising.
    - Script: Create an AOE slow. Public variables include; Area of effect, slow amount, animation/delay time.
- Shovel
    - Description: This is another animation use only object. The Shovel is used to whack another player at melee range to do 2 damage to them. It will only be on screen for a fraction of a second so keeping it very low poly is key.
    - Script: The player must be at melee range and when they press the ability button, there will be a collision check and if player gets hit, they lose 2 health. Public variables include; Damage amount, Area of effect
- Rapid-Fire
    - Description: An ability, it uses the same gameobject as the gun except a different script.
    - Script: Adds unlimited ball count to the player temporarily and increases fire rate. The projectiles it shoots are capped at doing only 2 damage to someone.
- Giant Snowball
    - Description: The Giant Snowball is for an eliminated ability, to summon a giant snowball on an area of the arena to slow the players. It's so that even when the players are

eliminated, they can still participate. It will be a more detailed and much larger version of the Snowball ability's Snowball.
    - Script: There is a bool to check whether they are eliminated or not. This ability is only available if they are eliminated. The Giant Snowball drops a snowball on top of the player's reticle. They use a reticle to aim it. It is basically the same as a snowball ability but in a large AOE.
- UFO
    - Description: UFO(Unidentified Flying Object) is a second eliminated ability that pulls the players within an area towards the centre of the target area.
    - Script: There is a bool to check whether they are eliminated or not. This ability is only available if they are eliminated. The UFO pulls players toward the player's reticle. They use a reticle to aim it.
- Canvases for UI elements
    - Description: The canvas system is how Unity uses 2D sprites for UI. There will be a few used for any UI we may need such as health or pause menu.
    - Scripts: The canvases are the UI elements. The script depends on which UI element it is. See UI section of TDD.
- Ambiently Snowy Objects
    - Description: These will be objects around the edge of the backyard to fill out the space a bit more and make it look more like a backyard. One example could be a dog house that has some snow on the roof.
    - Scripts: The snowy objects are purely for aesthetic and won't have any code.

Characters

- Player Characters
    - Description: These are the playable characters. They are children that will be in a cartoon-like style and will be a bit rugged up as they are still playing with snow which is quite cold.
    - Script: Player can move around by running using either WASD or the analog stick of an Xbox controller. They will have a capsule collider taking up majority of their body. They are able to shoot the tennis balls if they have collected one. Players are able to 'dash' which will move them forward at a higher velocity than running. It will have a quick cooldown so that it allows for a lot of mobility. There will be a boolean to see whether they are mid-dash or not as you can dash into a moving Tennis Ball to pick it up instead of taking damage. This boolean can simply be set to true while dashing.
- Player's Guns.
    - Description: The way that the players shoot the Tennis balls at each other. The guns will shoot the Tennis ball in the direction they are facing(their local forward direction)
    - Script: This will be parented to the Player Character. The Player Character will handle the shooting code so the gun won't have any code on it.

Input Method

Keyboard and mouse input for PC & Xbox Controller.

Function	Keyboard	Xbox Controller
Menu and Pause	Esc	Start
Menu Navigation	Esc and Mouse left click	Left Analog Stick and D-Pad
Select	Mouse Button 1	A
Back	Esc	B
Movement Up	W	Left Analog Stick
Movement Left	A	Left Analog Stick
Movement Down	S	Left Analog Stick
Movement Right	D	Left Analog Stick
Dash	Space Bar	Left Trigger
Aim	Mouse Movement	Right Analog Stick
Main Throw	Mouse Button 1	Right Trigger
Primary Eliminated Ability	Mouse Button 1	Left Bumper
Ability	Mouse Button 2	Right and Left Bumpers
Secondary Eliminated Ability	Mouse Button 2	Right Bumper

User Interface

The menu will be controlled by having a selected button and changing selected button by using the same way as movement in the game. It is WASD on keyboard or Controller left analog stick/ directional pad. Once you have chosen which button, press ENTER on keyboard or A on controller. To go back to the last menu screen you can press ESC on keyboard or B on controller.

Health will be displayed on the screen during gameplay as a number of how many more hits until they're eliminated. The way that this will be done is; Take the player's current health int and display it on the UI as a number for the players to see.

The pause menu will be brought up by pressing ESC on the keyboard or "select or start" on controller. When it is brought up, it will pause the scene. It will pause the scene by setting Time.timeScale to 0. It will also bring a canvas over the top of the scene. It will have a low-opacity grey over everything and have a menu pop up in the centre. Due to timeScale becoming 0 everything will pause and you can choose what you want on the menu.

In the character customisation menu, there will be arrows on the sides of each aspect that can be changed and you can have one selected section by going up or down using d-pad and analog sticks, on keyboard it is W and S. To toggle between them by right and left on the analog stick/d-pad. On keyboard is it A and D.

System Requirements

OS: Windows 7/8/8.1/10

Minimum Requirements:

- Processor: Intel i3 6100
- Memory: 4GB RAM
- Graphics: NVIDIA GeForce GTX 850M
- DirectX: Version 11
- Storage: 2GB Available Space

Recommended Requirements:

- Processor: Intel i5 4600
- Memory: 4GB RAM
- Graphics: NVIDIA GeForce 1050
- DirectX: Version 11
- Storage: 2GB Available Space

Third Party Tools

Unity 2017.1.0f3 (64-bit)

Xinput for Unity (Package)

GitHub

Visual Studio

TortoiseSVN

Hack ‘N’ Plan

Discord

Photoshop

Maya

ZBrush

Substance Painter

XNormal

Marmoset Toolbag

Audacity

Shader Forge

Probuilder Basic

Scene OBJ Exporter

Coding Conventions

Capital function names: Example, Stunned(), KnockedOut() or Idle().

Hungarian notation: For example a bool controlling the players alive status would be ‘bAlive’, as for integers they use ‘n’ instead of ‘i’ so the health value would be ‘nHealth’.

More information here: https://msdn.microsoft.com/en-us/library/aa260976(v=vs.60).aspx

Camel Case: Every word starts with a capital in function names and variable names, used in conjunction with Hungarian notation. Example bool for if a player has the ‘big gun’, ‘bHasBigGun’. All variables declared in the main class scope will start with “m_” to denote them being members of a class.

Source Control

Program - TortoiseSVN

Branches will not be used

Version numbers, increment by 0.01, start at 1, and be appended with ALPHA or BETA, example first commit version ALPHA1.01, second commit version ALPHA1.02, then ALPHA1.99 -> ALPHA2.00. when we start beta, BETA1.03 and BETA1.67, and so on for GOLD.

If nothing has been implemented alphabetical numeration will be used: ALPHA1.09a, and ALPHA1.09b for whem you're just tinkering.

Team Members

Nathan Nette - General Programmer

Mitchell Cattini-Schultz - General Programmer

Marc Varvakis - General Artist

David Condello - General Artist

Matthew Franzi - General Artist

Hamish Smithers - Designer (Producer)

Emma Wearing - Designer