Shifty Duels [tbc]

<1.0>

Konstantin Schaper  
David Kuehlmann  
Alexander Hoffmeister

Video Game Design (CGP603)

Unit Lead: Christian Brindley

**DELETE SECTIONS IF THEY DO NOT APPLY TO YOUR GAME E.G. MONETIZATION  
(delete this message before hand-in)**

[Revision History 4](#_Toc464116256)

[Concept Document 5](#_Toc464116257)

[Game Goals 6](#_Toc464116258)

[Story Overview 7](#_Toc464116259)

[Game Controls 8](#_Toc464116260)

[Technological Requirements 9](#_Toc464116261)

[Front End of Game 10](#_Toc464116262)

[Title/Start Screen 11](#_Toc464116263)

[Game Flowchart 12](#_Toc464116264)

[Loading Screen 13](#_Toc464116265)

[Game Camera(s) 14](#_Toc464116266)

[HUD System 15](#_Toc464116267)

[Player Character(s) 16](#_Toc464116268)

[Player Metrics 17](#_Toc464116269)

[Player Skills 18](#_Toc464116270)

[Players Inventory Tools 19](#_Toc464116271)

[Combat 20](#_Toc464116272)

[Power-up/State Modifiers 21](#_Toc464116273)

[Health (if applicable) 22](#_Toc464116274)

[Scoring (if applicable) 23](#_Toc464116275)

[Rewards and Economy 24](#_Toc464116276)

[Collectibles/Object Sets 25](#_Toc464116277)

[Vehicles 26](#_Toc464116278)

[Game Progression Outline 27](#_Toc464116279)

[World Overview/Level Select/Navigation Screen 28](#_Toc464116280)

[Universal Game Mechanics 29](#_Toc464116281)

[Game Levels 30](#_Toc464116282)

[General Enemy Rules 31](#_Toc464116283)

[Level-Specific Enemies 32](#_Toc464116284)

[Bosses 33](#_Toc464116285)

[NPCs 34](#_Toc464116286)

[Minigames 35](#_Toc464116287)

[Monetization 36](#_Toc464116288)

[Downloadable Content 37](#_Toc464116289)

[Cutscenes 38](#_Toc464116290)

[Music and SFX 39](#_Toc464116291)

[Other Screens 40](#_Toc464116292)

[Appendix(es) 41](#_Toc464116293)

# Revision History

Update this information with publishing dates and track which author updated the material

# Concept Document

**Game Name:** Shifty Duels [tbc]

**Genre:**. Turn-Based 1on1 Arena

**Description:**

Shifty Duels is a turn-based, one versus one arena game that is played on a square grid.

The game either ends by one player dying or when a set number of turns have been played. The players gain points for dealing damage to the opponent, healing themselves or taking secondary objectives. The player with the most points at the end of the game wins.

The matches take place on floating debris in space with varied terrain. At the end of every turn the terrain randomly shifts in height, so the player has to adapt on the fly.

**Game features:**

* 1vs1 matches
* Isometric on a square grid
* Constantly shifting surroundings
* Variety of characters
* Interesting array of abilities for each character
* Different ways of winning the game

**Resources:**

* Unity (+ MonoDevelope/Visual Studio)
* Photoshop
* 3dsMax
* Audacity
* Trello
* GitHub (+ SourceTree)
* MS Office

# Game Goals

Our game is a turn-based multiplayer arena, the goal is to either kill your opponent or to have the most points at the end. The players spawn on opposing sides of the maps. Both have a hero that has several skills at their disposal, clever use of those skills is the key to victory.   
The game will be played on a square grid of tiles and the main feature of the game comes into play at the end of each turn, at this point the height of the individual tiles changes changing the look of the arena.   
As the game starts both player will find their heroes on opposite sides of the arena. Each player will have a turn consisting of moving his character and attacking his opponent if he wants to or is able to.  
To do this he has an array of four different abilities, that can target the environment, himself or the enemy. Use of these abilities consumes resources and they will incur a cooldown.   
After both players had their turn, the height shift comes into play. This mechanic will change the height of the individual tiles of the arena. This may block line of sight need for abilities or attacks or open new ones.   
The game will end when one player kills the opposing players hero, thus winning the game.

The game will be developed in Unity and be playable on PC.

# Story Overview

No direct story involved in the gameplay is planned for the game. The game is solely a multiplayer arena and does not focus on storytelling in any sort of way.  
A short “backstory” of the arena would be the following:

The setting is an abandoned space station that has accumulated a lot of space debris. While no organic life exists here, old robots from the stations golden days still roam the it. These robots fight for resources to maintain their mechanical bodies. Made all the harder by the ever shifting properties of the debris and the haywire leftover controls of the station.

# Game Controls

The game is played on a square grid. Everything can be controlled with the mouse, key bindings for abilities and menu.

Movement is done by clicking the character then choosing move and clicking where you want to move inside the shown move radius.

After that the player can use skills, by either using a hot key or clicking the skill, every skill has a radius where it can be used.

# Technological Requirements

What tools will this game use?

How are the camera, physics, bosses and so on going to be done? Implemented by programmer? By Designer? Hard-Coded? Scripted?

What design tools will this game use? List level creation and scripting tools used.

What are the proposed tools for cheats? Include controls for level, invulnerability, camera, and other gameplay-related cheats

For PC games, include a section covering the target specs of the computer needed to run the game. This will list things like the amount of RAM, minimum CPU speed, minimum graphical abilities, required by peripherals, and so on.

The game will be done in Unity.

Unity standard speccs

# Title/Start Screen

**Start screen:**

The start screen will have a number of option these are: Play, Options and Exit

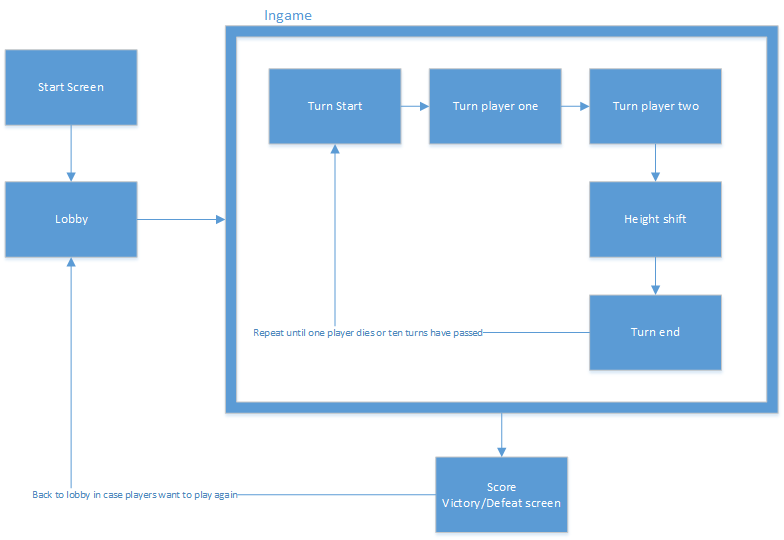
**Options:**

The options provide the possibility to change various game settings such as screen resolution and adjusting audio levels.

**Play screen:**

Here the player can choose his hero and host or join a custom game.

# Game Flowchart



# Loading Screen

The Loading screen will, where it is needed, show an illustration of the arena.

# Game Camera(s)

The camera will be isometric, top down, focused on the map rather than being focused on the players hero. It can be centred on the hero by tapping “SPACE”. It can be moved by using the arrow keys or moving the mouse to the screen borders.

# HUD System

What we need to show:

* Score

The HUD will contain a number of elements necessary for the game. Since the player control a hero it is imperative that they know the status of their hero at all times. As such we will have displays for health, action points and skills. All of this will be in the bottom centre of the screen.

In addition the player will be provided with a mini map of the arena. It will be located in the bottom right corner.

# Player Character(s)

There is no singular player character instead the player can choose a predefined hero at the start of each match.

# Player Metrics

List and detail the player character and provide metrics pertaining to movement, combat, context-sensitive moves (such as QTEs), health, player death, and idles.

# Player Skills

Since we do not have a single player this will be a list for a prototype hero.

Turret Hero [development name]:

Skill 1:

Place a turret, within a defined radius, on the map.

If the turret is already placed move it to a new location instead.

Skill2:

Both the character and the turret shot their weapons at the targeted location.

Skill 3:

Pushes all other characters away from the character and the turret.

# Combat

Describe and illustrate all combat moves and reactions including combo moves; different weapon types (melee and ranged); weapon tech tree; ranges; how the player equips; reloads; and changes weapon; lock-ons; and targeting system.

# Health (if applicable)

Describe how the health is tracked on the HUD and how players can lose and replenish health. Describe how players can tell when health is low.

**Alternate states**—describe any alternate states (stunned, poisoned, turn into a baby, ect) the player can get into and how it might affect controls.

**Lives** (if applicable)—explain how lives are earned or lost and what happens when the player runs out.

**Death** (if applicable)—describe what happens when death occours. List situations requiring unique animations (fire, drowning, and so on). What happens when the game is over? What does the game-over screen look like? Is there a penalty for dying?

**Checkpoint System**—Describe the in-game checkpoint system. How does the auto-save system work?

# Scoring (if applicable)

Assign point values to actions and explain what happens when players reach them. How do players earn bonuses in the game (like chaining or combo)?

**Leaderboard setup**—what does it look like? What stats are being tracked?

**Achievements**—what achievements are available and how are they earned? List them and provide images for badges if applicable.

# Universal Game Mechanics

List mechanics that will be found throughout the game. Always include images of each mechanic. List each platform, portal, breakable, hazard, intractable object, and puzzle element and how the player interacts with them.

# Game Levels

The will be one level which will be the arena. It would be a derelict space station and debris.  
The galaxy and a planet as a backdrop. Music should be mainly in the background. Soundeffects need to be clearer, as they underscore the abilities.

# Monetization

The game will be a one time purchase. Further money could be made by selling DLCs containing new maps and heroes.

# Music and SFX

List all music needs. Describe the tone or feeling of each piece. List on what level the music is needed, and don’t forget title, pause, and option screens as well as end credits.

How the fuck should I know

# Appendix(es)

This is the place where long lists go, including player animations, enemy animations, sound effects, music, cutscenes scripts, in-game text, and VO scripts.

This is where your task list should be placed.

This is where your Asset list will be.

You should break this section up in a readable manner using sub-headers ect.

Place list of textures needed.