By Early 0'Clock Productions

Technical Design Document V4.1

Champlain College

May 2, 2019

2019 Ubisoft Game Labs Competition

Index

Click to jump to section

Overview

Game Concept

Descent of Champions is a round based arena brawler where players must compete to win the audience's favor. Spectator's watch the game live and influence gameplay to make their show more entertaining. Players in the match will complete objectives and adapt to the changing arena to gain the most points and win the game.

Delivery Platform

PC release is the target platform as it is the most common streaming platform but since the game is meant to be played with a controller other platforms would potentially be supported. The website will be designed for mobile but will work on desktop as well. Major brand mobile phones that have been released in the last 2 years are optimized to work with the site. The target stream platform we are integrating with for the prototype is Twitch but for full release would be able to support other large platforms such as YouTube.

Technical Goals (Current)

Top Priority:

- Implement announcer voice line systems
- Bug fix issues found during Ubisoft playtest
- Implement *Revenge of the Grunts* DLC
    - Exploding grunts audience buyable
    - Visual queues to show event has started
- Implement hats and custom audience colors

Secondary Priority:

- Implement Confetti mode - everything explodes in confetti
- Implement Easter Eggs because why not
- Implement website art for DLC buyables
- Implement bonus in game art

Art Pipeline

Art will be imported into the game in the existing file structure with the naming convention **DoC_nameOfObject_typeOfObject_specifier.fileExtention**. Importing art from Maya will be done to an external folder to assure files are named and

Back to Top

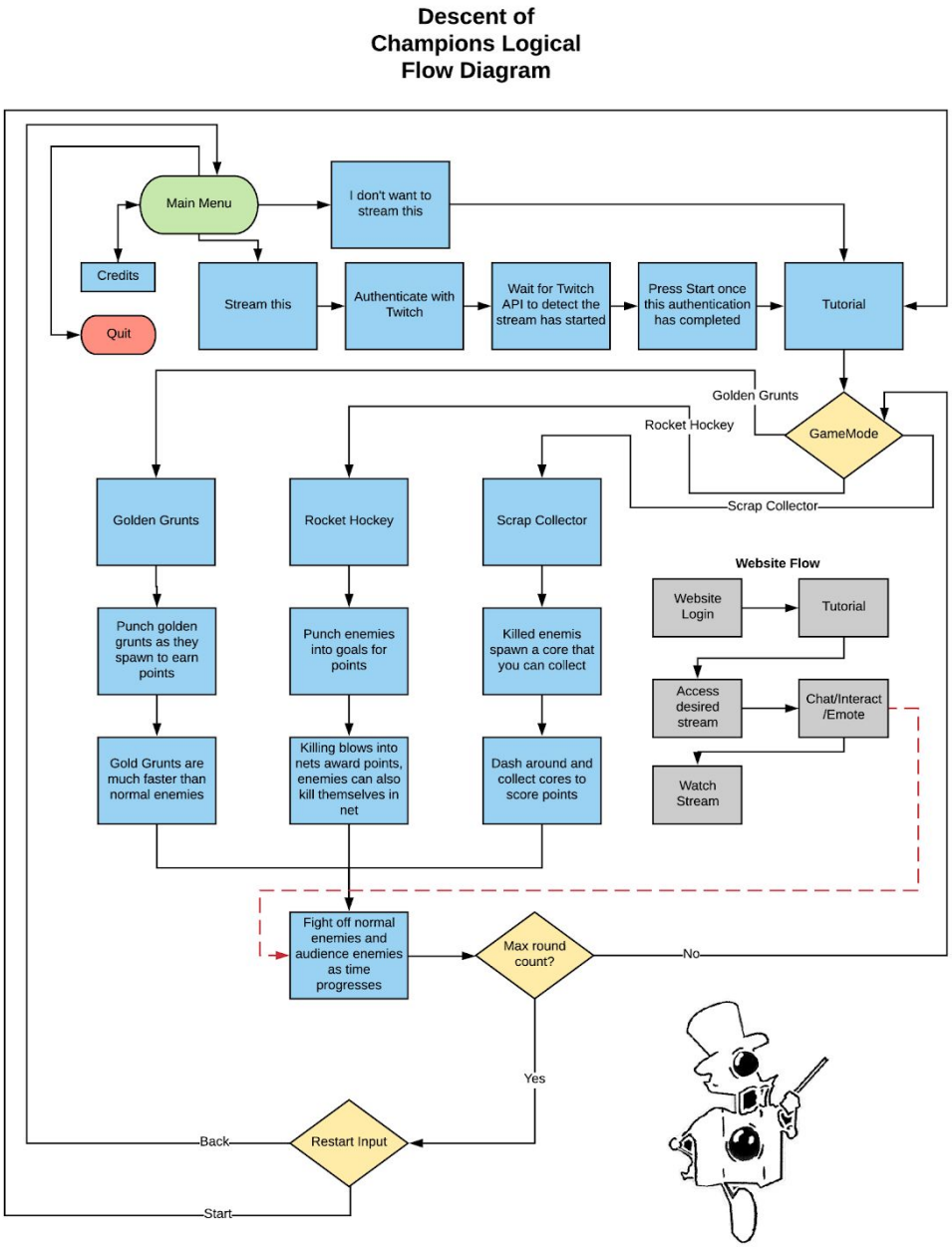
formatted correctly and then moved into the Unity project in the correct location. Art will be committed to the repository in it's own branch to be merged and implemented by programmers later. Artists are responsible for checking in engine that assets work as specified. Artists will also be able to apply textures, animations and shaders in engine for working directory branches with help from programmers. Artists will refer to the GDD for specifics on functionality of art and animations to ensure they will work with existing systems. Art will also be planned one week in advance from implementation so programmers do not get held up in implementation. For technical assets (assets that may need pieces to be moved individually) ensure that when exporting all objects in the parent have the ability to be moved freely relative to the position they are in world space rather than to the parent as well as making sure those assets are individually accessible pieces and not one large mesh under the parent.

Design Pipeline

Everything for programmers to implement per milestone should be referenced in the GDD in detail. Every sprint the GDD will be updated to reflect tasks of the coming week. After first implementation of features designers will balance in engine and only call on programmers to revisit features for reworks when massive changes are necessary. Programmers are responsible for implementing features in a way that allows designers to easily modify conditions and variables through the inspector. File structure is to be followed for additions in engine and good code architecture is expected if scripts are added by designers (See Microsoft C# Coding Standard for more). If features come up in the middle of a sprint they must be held until a new sprint as to not overburden programmers with excessive expectations per sprint. Designers will be planning one week in advance from implementation so programmers do not get held up in implementation. Andy is lead designer for immediate decisions. When possible ask relative designers to their specialty but if decisions are being held up and need to be answered Andy is in charge of making those decisions as a lead. Programmers and Artists must read the GDD in detail every release to keep up to date on the latest changes. Designers will also give a list of public variables they wish to have accessible in a feature in their design documentation so Programmers can plan accordingly. **For programmers and designers in engine you should make a copy of the most recent build scene and do edits in the test scene. When the work is complete you can add those changes to the main scene and assure that you only affect what you need in the main build. DO NOT commit test scenes unless you need to come back to work later at a different location. You should have completed the task to the fullest extent and implemented into the game in a FINISHED STATE.**

Back to Top

Logical Flow Diagram Find the full PDF here.



Back to Top

System Requirements

Required:

- Game:
    - 3 Xbox controllers
    - Windows 7 and up
    - 2nd generation Intel Core i3, AMD - FX-6300 6-Core Processor, or higher
    - 4 GB RAM
    - Graphics: Nvidia 8800 GT, AMD 5670, Intel HD 3000
    - 1.0GB of storage
- Website (Spectator):
    - All smart phone brands
    - Model year 2010 and up
    - Any browser
    - Desktop accessible

Recommended:

- Game:
    - 3 Xbox controllers
    - Windows 10
    - 7th gen Intel Core i5 or AMD equivalent
    - 8 GB RAM
    - Graphics Nvidia 1070 or AMD equivalent
    - 1.5GB of storage
- Website (Spectator):
    - iPhone or Android
    - Model years within the last 2.5 years
    - Google Chrome

Systems

Movement:

Minimum controller dead zones are utilized as controls should be snappy and instant. There is still an acceleration to the speed of the player through percent of controller stick input but there will always be some form of input. Button interactions are a mix of holds and presses, special moves have held aims/charges where movement and small attacks are instant presses.

Back to Top

Physics:

The game removes as many random physics components as possible in the controls and enemy interactions but maintains a random physics element when enemies die. The enemy parts detach from their parent object and an explosion force is applied to them. Collectibles are also affected by physics. Forces are being monitored on most objects to ensure if an object is not moving when it should that it is forced to comply. This system allows for some control over the randomness of the physics engine.

Levels:

Every level takes place on a floor composed of hexagons that can be set to be used as spawn points for enemies. Objectives for a level are spawned independently of the floor. Objectives are to be textured similarly to indicate that they are objectives that give the player points such as gold hockey nets, golden grunts, and golden scraps for scrap collector. The floor breaks apart at the end of a round and explosion force is applied to the pieces similarly to how enemies are treated. See **Physics** above. Players fall through the floor along with audience spawned enemies and objects and are placed in position once off camera for the next round.

AI:

The AI is controlled by NavMesh. Different behaviors are programmed for each enemy following a base class to ensure the similar traits are kept (death animations are shared between enemy types by exploding to give an example). Downsides of the NavMesh component is the floor is not static for the whole game and enemies tend to leave the ground at times. This can cause errors from the NavMesh by not having a valid target. To resolve this we do frequent checks to ensure the AI only checks to move when possible and conditions for the NavMesh agent are satisfied.

Win Condition

Players can gain score through different means on the different objectives. The score is cumulative through rounds up to 5 rounds. At the end of the rounds the person with the highest score (denoted by a crown UI element during the game) is declared the winner. If players tie, the winner is decided by their cumulative objective scores between the rounds. If players tie in round and objective scores, the winner is decided by player ID. This scenario is very rare but accounted for.

Back to Top

Code Overview

Development Environment

Unity 2018.3.4f1 is the target build for this game. Packages implemented at the beginning of the project may be dependent on version 3.0 so it is not advised to use older builds. Rewired is being used as the main controller input manager. Mercurial is the chosen repository manager with Pineapple hosting the main repo. Builds of the game must be made using version 2018.3.4f1 preferably within 1 version forward to maintain consistency. 2018.3.4f1 solved a bug that caused issues with rasterization resulting in a game crash when the error was created.

File Formatting

Code is written in C# (.cs) and conforms to the **Microsoft Coding Standard** for in engine systems. Website implementation is written using PHP and Ratchet for websockets. HTML, CSS, Ajax, Javascript for website front end. The server uses Apache and MySQL tables. Naming convention for all script files should be as follows:
functions - FunctionName(); variables - variableName;

Comments

Comment all new code you add to effectively communicate to the team what each addition does so this document can be updated accordingly, and other members can work without needing a decoder ring. Include any specific comments about unfinished code or anything else that may need explaining clearly in the files changed.

Source Control

Using Mercurial through Tortoise Hg to manage repository. Pineapple is hosting through Champlain College. Find out how to set up Mercurial here. Currently server information is stored on a private GitHub. Implemented interaction code is kept with the main game repository. A Microsoft Azure server hosted the website leading up to the competition date (April 1, 2019) but from that point forward is set up to work on local machines connected to the same network.

Back to Top

User Interface

Game Menu

Twitch authentication for the streamer will be accessible when starting the game. This takes you to a webpage to verify with Twitch that your account is correct so streams cannot be created by fake users under a popular name.

Website

Login screen that is prompting for username. Brought to lobby to show active streamers. After selecting a stream you are brought to the stream with an active chat. Other tabs include “shop” and “emote”. The shop allows users to crowd fund in game interactions and currency is earned through passively watching the stream and by tracking participation on the website. New tabs added after the competition date (April 1, 2019) include an emote customization system that allows you to change the color and hat of your emote in the game.

Task List

Current Milestone (DLC):

- Add announcer voice lines and system to play them (Gameplay)
- Add announcer voice lines (Flavor commentary)
- Bug fix issues found at Ubisoft playtest
- Implement backend for hat and color customization systems
- Implement *Revenge of the Grunts* DLC content
    - Grunts explode after first attack
    - Grunts smoke to denote that they are enraged
    - Implement UI to denote event has started
    - Implement art to denote grunts are different from regular grunts
- Implement Confetti Mode
    - All enemies explode with confetti after death
    - Just… confetti everywhere
- Polish existing systems and implement QA feedback

Back to Top

Technical Risk

Website - Hard

The website is by far the biggest risk to this game but it is also the best way for us to get meaningful player interactions in a way the is easy to pick up, fun to use, and give us complete control over the experience. Server implementation and scalability will be difficult to implement as well. Twitch plug-ins would be a safe fall back as Twitch is very willing to get developers making new interactions for their platform and this would be easier in the event that the website crashes and burns somewhere down the road. Programmers in the dev team have very limited experience with web development but with the progress made in the early stages of prototyping we are comfortable accepting this risk seeing the progress that has been made.

Team Size - Medium

This many people in a project makes it difficult to manage a clean and working repository as well as maintaining effective communication. Through the use of this document we hope to minimize this risk and ensure all developers are on the same page as well as have effective knowledge of repository use as to prevent major impediments in the future.

Testing Feedback - Medium

Due to the limitations of QA we have limited fresh views to test our game with along with limited time to test in the Montreal QA. We have to reach out to people beyond our school to get fresh views which could be difficult but luckily we are in Montreal and finding willing participants should be possible.

Unity Packages - Medium

There are packages in our Unity build to make life for artists and designers easier such as HD Render Pipeline that programmers have never worked with before and have already caused some minor set backs in the prototyping phase. We believe that despite the initial setbacks it will help later in the development process so we accept this risk in hopes for an easier life down the road.

Addendum (4.15.19) After significant development with HDRP in the project it has come to our attention that the package is far too early in Alpha to be deemed useable for large scale projects and the initial risk associated with the packages was not properly measured. Numerous setbacks have resulted from the HDRP included in the project but it is too far into production to remove the

Back to Top

package. From this point forward the team has become quite familiar with the limitations of HDRP and it would be more risk to remove the package than to live within our limitations.

Art Pipeline Risk - Medium

Our programmers have never worked with 3D artists before as well as have limited experience with shaders and graphics in general. Working with 3D artists will be an initial challenge but a completely necessary one as the game is in 3D and there is no getting around that.

Team Sign Off

James Van Nuland_____
Producer

Scott Aquino _____
Programmer

Mark Botaish_____
Programmer

Tyler Chapman_____
Programmer

Andersen Pinckney_____
Designer

Richard Hardy _____
Designer

Nicholas Kline_____
Artist

Isaac Mills_____
Artist

Back to Top