Game Design Document

Johnny’s Revenge

COMP7903 -Game Design Fundamentals

COMP7051 – Intro to Computer Games Development

December 6, 2020

Team RJSJ Members

**Richard** **Tesch**

**John Dong**

**Shaun Tseng**

**Jaedin Dhatt**

Table of Contents

[Design Team 4](#_Toc57756683)

[Methodologies & Tools 5](#_Toc57756684)

[Methodologies 5](#_Toc57756685)

[Tools 5](#_Toc57756686)

[Concept Document 6](#_Toc57756687)

[Game Rules 6](#_Toc57756688)

[Operational 6](#_Toc57756689)

[Constitutive 6](#_Toc57756690)

[Implicit Rules 6](#_Toc57756691)

[Core Loop 6](#_Toc57756692)

[Discover 6](#_Toc57756693)

[Fight 6](#_Toc57756694)

[Deck Improvement 6](#_Toc57756695)

[Core Mechanics 6](#_Toc57756696)

[Cards 6](#_Toc57756697)

[Opponents 6](#_Toc57756698)

[Deck 7](#_Toc57756699)

[Hand 7](#_Toc57756700)

[Mana 7](#_Toc57756701)

[Health 7](#_Toc57756702)

[Card draw 7](#_Toc57756703)

[Rewards 7](#_Toc57756704)

[Drag/Drop 7](#_Toc57756705)

[Notable Mechanics 7](#_Toc57756706)

[Timer 7](#_Toc57756707)

[Score 7](#_Toc57756708)

[Stage 7](#_Toc57756709)

[Level 7](#_Toc57756710)

[Signifiers 7](#_Toc57756711)

[Shield 7](#_Toc57756712)

[Sound 7](#_Toc57756713)

[Animation 7](#_Toc57756714)

[High Score 7](#_Toc57756715)

[Bonus 8](#_Toc57756716)

[Analysis of Dynamics 8](#_Toc57756717)

[Battling 8](#_Toc57756718)

[Mana strategies 8](#_Toc57756719)

[Rewarding 8](#_Toc57756720)

[Scoring 8](#_Toc57756721)

[Game Aesthetics 8](#_Toc57756722)

[Sound Design 8](#_Toc57756723)

[Visual Design 9](#_Toc57756724)

[UI components 9](#_Toc57756725)

[Intended Target 10](#_Toc57756726)

[Player Profile 10](#_Toc57756727)

[Persona 10](#_Toc57756728)

[Interest Curves 10](#_Toc57756729)

[Decision Trees 10](#_Toc57756730)

[Game Design 10](#_Toc57756731)

[Software Design 13](#_Toc57756732)

[ALPHA 13](#_Toc57756733)

[Prototypes 15](#_Toc57756734)

[Playtest 15](#_Toc57756735)

[Design Flaws 16](#_Toc57756736)

[ALPHA 16](#_Toc57756737)

[BETA 16](#_Toc57756738)

[RELEASE 16](#_Toc57756739)

[Next Steps 16](#_Toc57756740)

# Design Team

A list of the team members and their contact information (student emails work) ***John***

# Methodologies & Tools

To design and develop Johnny’s Revenge the team encompassed certain methodologies and tool sets to enforce parody, reproducibility and assure success.

## Methodologies

Given the nature of the environment the team will work in to complete the game it was recognized that we needed a development solution that would provide quick requirements and accompanying solutions through cross-functional members. The environment would have to support quick informal but productive collaboration of members to iteratively produce a package that would meet the requirements of the project.

The team concluded the best methodology is to use the iterative waterfall method to tie us to the milestone set out in the requirements. For building our requirements and developing solutions the team adopted Agile practices. Using short sprints of once a week and progressing to 3 times a week near milestones or end of project we can complete the project with success.

In the document folder the team has recorded all sprint activity in the agile folder and include a link to Trello used for tagging assigned sprints.

TODO: SUBMISSION PACKAGE: Agile/Sprint folder containing one or many documents for the sprints. Including dates, agenda, sperate Trello link.

* Trello Link - note too busy so we stopped at the beginning of November (https://trello.com/b/8JgkpTTS/project-management)

## Tools

* Discord – Meeting platform for all online meetings and discussions
* Unity – Free cross-platform game design platform.
* Blender – Used for 3D modelling of objects used in animations and reward screens. All used within Unity.
* C# – The language used to script objects within Unity
* Target platforms – Initial release only on Windows platform. Future targets after release are the mobile market.
* Version Control – GitHub repository using WEB and Git Desktop for access. Use of developer branches to develop and test assigned features before main code base merges.
* UML – Diagram of Implementation Classes, Hierarchy, Package Template, Package Diagram, Type Hierarchy, Game Design
* Testing – all developers playtest, play testers, and bugs recorded in sprints
* Bug fixing - pulled from sprints and corrected before next sprint

## Concept Document

***Shaun*** *– formalize use WORD heading for table of content to pick up*

* + Sketches
  + Our concept doc from presentation. Use the new core statement from this document.

## Game Rules

**(WIP) John**

### Operational

### Constituative

### Implicit Rules

* + Card tables with values and stats ***(HAVE write up each card?)***
    - Player
    - Reward
    - Enemy

## Core Loop

***Jaedin (we HAVE need explanations)***

### Discover

### Fight

### Deck Improvement

## Core Mechanics

***Shaun***

### Cards

### Opponents

### Deck

### Hand

### Mana

### Health

### Card draw

### Rewards

### Drag/Drop

## Notable Mechanics

***Richard***

### Timer

### Score

### Stage

### Level

### Signifiers

### Shield

### Sound

### Animation

### High Score

### Bonus

## Analysis of Dynamics

***Shaun (REVIEW)***

### Battling

### Mana strategies

### Rewarding

#### Stage rewards

#### Level rewards

#### How are they calculated

#### How does one select

#### How it ends the dialog view

### Scoring

#### How are they calculated

#### Stage and running game total

#### High score

## Game Aesthetics

### Sound Design

***John*** List

* where sounds is used
* Credits
* Brief description(why chose it)
* Pirate themed

### Visual Design

***Jaedin*** ***(What they are, how they are used)***

#### Theme

##### Artwork

##### Pirate theme

#### Scenes

(literally unity scenes and any dialogs)

##### Menu scene

##### High Score

##### Adventure Map

##### Battle Screen

### UI components

#### Health

#### Mana

#### Shield

#### Buttons to select

#### Status visuals

##### Level location

##### Timer

##### Signifiers

##### Card Deck

#### Score / Reward Dialog

Bar % for health and mana reward

Text to represent game results

Animated Chest to associate reward

Buttons to select the reward after stage

Drag and Drop cards after level

## Intended Target

***John – brief introductory blurb (REVIEW have but needs formalizing and write up. I.e. research done persona and profile, any other analysis, how type and game are compatible)***

Everything from previous presentations

### Player Profile

Persona and Analysis

## Interest Curves

***Richard (enemy progression plus card reward)***

* Interest graph for each level
* Flow zone
  + Show each stage and how the progression increase
    - MaxHealth = levelNum \* 10 + (levelNum - 1) \* 5 + stageNum\*10;
  + How do we represent the deck building in the flow zone?

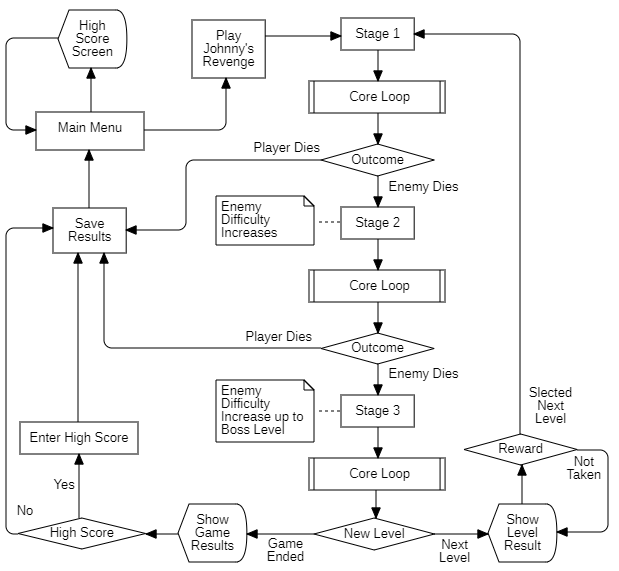
## Decision Trees

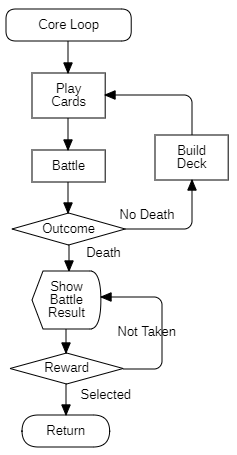
Our game design did not use decision tress as it is linear in design.

We have discussed this earlier in development and agreed that future development after release will build upon a string multi level adventure map that includes story telling. It is at this time that decision trees would be incorporated to allow for variations in choices provide stronger agency and increases immersion and habitation in the magic circle.

## Game Design

***Richard intro and explain the diagram***





## Software Design

***Richard***

Our software design was captured using UML to produce the necessary packages and subsystems to deliver the mechanics, dynamics, and aesthetic control.

The designs for each phase Alpha, Beta, and Release are recorded here are the

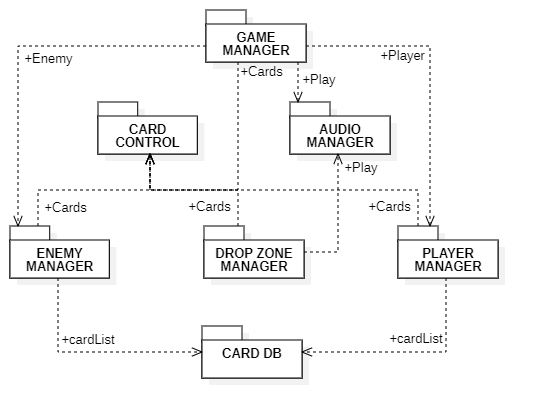
Package diagram: used to capture the high-level overall design of the system

Package template: used to start drilling down by showing the class block (no details) and how they perform simple interaction but only within a package.

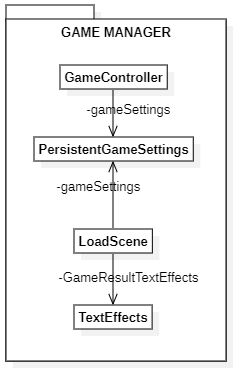
Diagram of Implementation Classes: Lastly, what UML traditionally called the class diagram. This is the meat and potatoes of each class and how the layered interaction between each other occurs. As it is of the highest granularity it also evolved the most over time and some spilled over into new classes of sub systems as they became to bloated or obtained cross purposes.

### ALPHA

Johnny’s Revenge : Model : Package Diagram



Johnny’s Revenge : Model : Package Template Diagrams :



#### Game Manager

Introduces the concept of an evolving game manager that will take care of all the higher-level details but all in one location. This include various support classes to help meet the package needs. It was decided that this effort of design early on would keep growth compartmentalized but time will tell.

##### Game Controller

Main class used as the conductor to orchestrate and coordinate all other packages to follow the constituative rules and some main core mechanics. For example, turn control, deciding when turns end, begin, or battles occur, scene navigation which includes the saving of persistent data to travel amongst scenes to share important values.

##### Persistent Game Settings

Classes only job is to have a collection of properties that can be abstracted from underlying platform storage. Once implemented it allows clients to manipulate hidden date at their needed layer of abstraction all the while making sure the data is present even when objects requiring it move out of scope or existence.

##### Load Scene

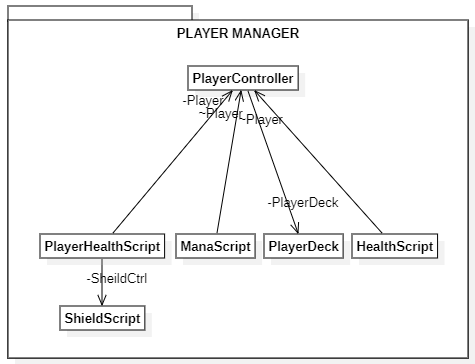
Created to handle movement between the only scenes restricted and controlled by the games design. When demands for switches occur it will take actions to assure that data and house tasks are dealt with before the switch.

##### Text Effects

The class is a cheap method to encapsulate some text effects such as fade, throb, flash or hide and show. Not a complete class as it may not be there in the final design but is a quick way to evolve what needs to be done with place holder affects.

#### Player Manager

Encapsulates the concept of delivering a manager that will deal with all the details of the player and its mechanics and dynamics.



## Prototypes

(Document all the prototypes that you will be making. There should be at least 1 per iteration)

***Jaedin (REVIEW have to pull out each iteration, screen shots and describe each change and what learned from it ) fill in the information below in yellow***

* Screenshots of each release
  + Shots must allow for single or combined demonstrate changes of
    - Mechanics
    - Dynamics
    - Aesthetics

### Playtest

***Shaun*** iterations and results first intro blurb then each milestone

* Format
  + - * How during proto, alpha, beta, & release
        + Surveys
        + Interview
        + What did each question try to answer
      * Results
        + pie charts
        + Where questions duplicated most recent result and then deltas
        + What it taught us

Past efforts

List beta and previous

Future goals

Each iteration leaving out release

#### ALPHA

#### BETA

#### RELEASE

## Design Flaws

***Jaedin (REVIEW Include ALPHA, BETA, & RELASE presentation playtest findings and what they drove. This is not actual playtest results as that is in the previous section. This is the “what did we learn summary” for flaws and if we took them on into the next release – documented in the “next steps” section.) Remember write an intro blurb of what is documented (probably after its done) and the follows the meat and potatoes you fill in.***

* + Insert and expand on the presentation” Future Scope”
    - Where they came from
    - What hope to achieve

### ALPHA

### BETA

### RELEASE

## Next Steps

#### ALPHA

#### BETA

#### RELEASE