**Taswell**

Game-Design Document :: Joshua Jefferis

*Hekit Software 2025*

# **High-Level Concept:**

## Title: Taswell

## Concept Statement:

*An action-platformer about a fox that must fight off waves of enemies while powering on elevators to advance to the next room.*

## Genre:

*2D-Platformer, Action, Side-scrolling Shooter, Rouglike*

## Target Audience:

*Fans of games like Risk of Rain 2, Enter the Gungeon and Hollow Knight; Teenager/Young-Adult Age Range.*

# **Product Design:**

## Player Experience:

*Taswell is designed to create a fast-paced, high-stakes experience where the player feels empowered, but constantly challenged.*

## Game World / Story:

*You are a fox trapped in the basement of an abandoned facility named ‘Taswell’.*

*You must re-activate the electricity-generators to power the elevators to ascend to the surface and escape of the laboratory*.

Monetization:

*Premium Game (Pay On-Launch)*

## Project Constraints:

* *Input: Keyboard + Mouse*
* *Game Engine: Unity 6.31 (C#)*
* *Language: English*
* *Prototype-Development Timeframe: 5 Days*

## 

## **Game Inspirations:**

**Gameplay:**

* *Risk of Rain 2 (2020)*
* *Celeste (2018)*
* *Enter the Gungeon (2016)*
* *Hollow Knight (2017)*

**Visuals (Artstyle):**

* *Noita (2019)*
* *Risk of Rain (2013)*
* *Katana: Zero (2019)*
* *Celeste (2018)*
* *Enter the Gungeon (2016)*
* *Scourgebringer (2020)*

**Setting/Environment:**

* *Portal 2 (2011)*
* *In Sound Mind (2021)*

## Visuals & Audio:

*Artstyle: Pixel Art*

*SFX: Original, Contextual, Heavy/Powerful*

*Music: Original, Dramatic/Cyberpunk*

## Style (Mood Board):

# **Assignment Specifications:**

## Client Criteria:

* *Must be FUN*
* *Must not have a rating higher than M (PEGI 17)*

## 

## MVP:

*Prototype for the first level of the game.*

## Highlighted Feature:

*The creations of a scalable Item system.*

## Highlighted Feature Description:

*A key feature of the game will be its roguelike-inspired random Item system; The player will always start off the same, but gain strength through the randomly selected stat-based items they find in the levels. The prototype of this game will be small-scale, but the game is planned to have a lot of items in the future. The problem is that I do not want to have to rewrite the entire code every time I want to add an item.*

*My solution is to create a modular and scalable item system that allows for easier production and testing of the game.*

## Solution Criteria/Checklist (Breakdown):

*The system needs to manage:*

* *Items affecting the Player’s stats (eg. Movement Speed)*
* *Multiple Items affecting the same Player Stat*
* *Stacks of Items (When you have more than one of the same item)*

*The system also needs to act in such a way that:*

* *It is as convenient as possible to create new items (Developer Convenience)*
* *The player can pick up and drop any number or type of item without the game breaking (Abstract Design)*
* *Some items can have linier effect, while others can be exponential*

## 

## Plan (Solution):

* *Create an “Item” Scriptable Object in Unity that contains all possible variables*
* *The player script will manage these Scriptable Objects in a List (Array), treating each item in an abstract way.*
* *When the player picks up an item, the player script will run a function that cycles through all the items and combines the variables into respective “multiplier variables”*
* *This “multiplier variable” will be constantly applied to the player’s respective variables in an Update method*
* *The function that adds the Item multipliers together will use if-statements to filter the exponential items from the liner items to handle the addition and multiplication.*
* *Items will be randomly pulled from a separate List of all possible items; This however will be a separate system that deals with RNG.*

## Solution Justification:

*I am happy with this solution because if successful:*

* *It meets all the criteria that I listed in the checklist*
* *The creation of items will be as simple as creating new instances of the “Item S.O.”*
* *This approach will make in-dev testing, experimenting and tweaking/balancing of items a lot easier do to the nature of S.O.’s*
* *Due to the power of mathematics, this multiplication/addition function method can also decrease stats by simple making items contain negative/decimal values*

## 

## Testing Procedure:

* *Run the game in the current state*
* *Focus on one element per test (eg. Movement, Stat Functionality)*
* *Use debug work-arounds to test functionality quickly (eg. manually assigning items)*
* *Make note of anything that is not working correctly*
* *Go back and fix those issues*
* *Repeat until system works as intended, checking against the criteria from before*

*Note: This is NOT bug-testing, this is just the cycle I will use to get systems working.*

## Feature Table:

| TESTING | EXPECTATION |
| --- | --- |
| Player Controller (ASWD) + Sprite Animations + Camera | Characters can move and jump around in the world responsively, with working animations. |
| Item System + Item Spawning | *Criteria from earlier in the document* + Items will spawn randomly from chests |
| Elevators and Power-Generators | Player can enter the map, exit the map, and power on the elevator |
| Adding Enemies + Attacking | Player can attack the enemies, and enemies can attack the player |
|  |  |

*Will use Google Forms for external closed-testing surveying.*

[*https://docs.google.com/forms/d/e/1FAIpQLSf6N8bmRIyV2UlgezXntqjISsXrarhzAM4ZEvuHZBtnOJiSeg/viewform?usp=sharing&ouid=110326384909001532625*](https://docs.google.com/forms/d/e/1FAIpQLSf6N8bmRIyV2UlgezXntqjISsXrarhzAM4ZEvuHZBtnOJiSeg/viewform?usp=sharing&ouid=110326384909001532625)