Strung Along – Development Doc

# Production and Timeline

## Definition of Responsibilities

Producer – Keeping team on track, co-ordinating team effort towards achieving goals. Making sure team members stick to the Gantt chart or reallocating resources if goals are not going to be met.

Character Controller – Designing and tuning the feel of the character. Coding different mechanics to enable character to move around and interact with environment.

Unity Support – Everything Unity: Adding assets from the artists, linking scripts to objects, working with camera, and defining global setup objects for the game.

Level Manager – Designing the script for transitioning between levels with specific attention to instantiating/revealing objects at the right time.

Level Design – Designing the puzzles that each level will contain. Creating an engaging difficulty arc that incorporates changing mechanics.

UI Programming – Designing the main menu as well as the settings panel where global game settings can be changed.

SFX – Capturing/procuring sound files to be used in the game. Designing what sounds should happen during what actions. Also responsible for engaging with the composer.

Character Design – Creating the 3D models of the characters and NPC’s.

Environment/Props Design – Designing the environment assets like platforms, backgrounds and objects that the characters can hold.

UI Programming – Designing the main menu as well as the settings panel where global game settings can be changed.

Particles – Designing particle effects.

Texturing – Giving all game objects realistic texturing, specific attention to wood and/or cardboard for most of the props and environment.

Rigging/Animation – Linking an animation frame to the character models as well as animating their different states of motion.

UI Programming – Designing the main menu as well as the settings panel where global game settings can be changed.

## Team Roles

|  |  |  |
| --- | --- | --- |
| **Name** | **Primary Role** | **Secondary Role** |
| Dylan | Producer | Character Controller |
| Tim | Character Controller | Unity Support |
| Harper | Level Manager | Unity Support |
| Andrew | UI Programming | Level Manager |
| Angus | Level Design | SFX |
| Skye | Environment/Props Design | Particles |
| Lauren | Character Design | Texturing |
| Dennys | Rigging/Animation | Level Design |
| Huw | SFX | Level Design |

## Tools Used

|  |  |  |
| --- | --- | --- |
| **Tool** | **Used for?** | **Reasoning** |
| GitHub | Version control and filesharing | Industry standard, flexible |
| Unity | Game engine, animation state transitions | Industry standard, powerful and easy to use, familiar to design team |
| Trello | Task Management | Clear and easy to use, recommended by supervisors |
| Autodesk Maya | Modelling, rigging, animation | Industry standard, powerful, familiar to design team |
| Substance Painter | Texturing | Industry standard, powerful, familiar to design team |
| Blender | Sculpting | Easy to use, excels at rapid prototyping |
| Photoshop | Texturing | Potential alternative to Substance Painter |
| Visual Studio | Scripting | Industry standard, integrated with Unity, familiar to programming team |
| Google Drive | Filesharing | Easy filesharing mainly for office documents, easy to work on at same time |
| Toggl | Time tracking | Recommended by supervisor, easy to use |

## Gantt Chart – Sem 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Timeline** | | | | | | | |
| **Events** | **Week 5** | **Week 6** | **Week 7** | **Week 8** | **Week 9** | **Week 10** | **Week 11** | **Week 12** |
| **Production** | | | | | | | | |
| Responsibility Allocations |  |  |  |  |  |  |  |  |
| Alpha Prototype |  |  |  |  |  |  |  |  |
| s |  |  |  |  |  |  |  |  |
| **Programming** | | | | | | | | |
| Player Controller |  |  |  |  |  |  |  |  |
| Moving Controller |  |  |  |  |  |  |  |  |
| Grabbing Controller |  |  |  |  |  |  |  |  |
| Level Manager |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| **Art** | | | | | | | | |
| Character Concepts |  |  |  |  |  |  |  |  |
| Basic Character Model |  |  |  |  |  |  |  |  |
| Stage Model |  |  |  |  |  |  |  |  |
| Prop Modelling |  |  |  |  |  |  |  |  |
| Advanced Character Models |  |  |  |  |  |  |  |  |
| Character Rigging |  |  |  |  |  |  |  |  |
| UV mapping + Texturing |  |  |  |  |  |  |  |  |
| **Design** | | | | | | | | |
| Act 1 Scenes |  |  |  |  |  |  |  |  |
| Act 2 Scenes |  |  |  |  |  |  |  |  |
| Act 3 Scenes |  |  |  |  |  |  |  |  |

# Mechanics and Design

## Overview

Two player couch co-op.

The players are puppets, however the strings connecting to each of their limbs combine into a single string, each anchored to a set point at the top of the screen.

The game is set on a marionette stage, levels are handled by hiding and revealing relevant objects and allowing them to fall or swing onto the stage.

The game camera remains in a fixed position overlooking a stage in isometric 3D.

When the players start, they are positioned inside the stage with fixed string-anchors to where they are attached.

Each level has a set objective which will involve puzzle solving to complete. There will be time pressure indicated by more and more hazards appearing as time goes on.

Upon either player’s death, they must respawn at the start of the scene.

## Level Management

|  |  |
| --- | --- |
| **Event** | **Details** |
| Level Start | Relevant objects will be “unhidden” and moved to their starting position (see Level Transition).  If the level is the start of a new act, the curtain will come up, there will be a fade/slight zoom-in to the stage. The players will start in their given positions. The objective may be highlighted or written on UI. |
| Level Complete | Upon completing the objective, the current stage objects will be moved off the stage (via force pushing), either upwards or into the stage “wings”. A Level Transition will occur.  If the level being complete is the end of an act, a triumph motif will be played, and the curtain will come down. |
| Level Transition | A critical component of the level flow is that upon the completion of a level, set pieces will be moved out of the scene, making the players drop to the ground if they were standing on top of them.  New objects will be unhidden and moved into position. If coming in vertically, they will be indicated with an oncoming shadow. If the players do not avoid objects falling in, they will be crushed. If coming in horizontally at speed, they may hit the players out of the way, sending them flying (subject to tuning). There is no impact damage unless the set piece is hazardous (ie. spiky or made of lava) |

## Player Control

|  |  |
| --- | --- |
| **Ability** | **Details** |
| Move | Standard 3D movement using force/momentum so that the character seems to ease in and ease out of movements. Responsive controls but medium speed, approx. 4 seconds for character to cross screen. Moving animation will play. |
| Jump | Standard force-based jumping. Strafing in air will be reduced to a 5th of movement on ground (subject to tuning). We will allow for late platform jumps by giving the players a moment of “coyote time”. Jumping animation will play.  No double jumps.  No fall damage.  Players will be able to hoist themselves onto ledges slightly above their jump height (see below). Hoisting animation will play.  If falling from height 3 times the height of the player, a “get up” animation will play. |
| Grab | Player will be able to grab onto environment objects and edges including player strings and the other player.  To hold a game object the grab button must be held down.  To climb (on climbable surface), the grab button must be held down while a direction input is pressed. Climbing animation will play.  Player can hang onto edges. To hang on a ledge, the grab button must be held. To hoist themselves up the ledge the “up” direction input must be pressed (while still grabbing). Hanging idle animation will play.  Releasing the grab button while holding onto something will cause the player to drop it immediately.  Bonus: We want to use inverse kinematics to make the player’s hand “lock onto” the object it is grabbing (think rungs on a ladder) |
| Die | Player will die if hit by anything defined as hazardous: spikes, enemies, tomatoes from audience. “Ragdolling” will occur instead of dying animation.  Player will die if crushed by falling set object. Crushing animation will play.  Player will die if falling into hazardous liquid: water, acid, lava etc. Drowning animation will play. |
| (Bonus) Attack | While holding handheld object, player can press attack button to swing object in enemy direction. Attack animation will play.  A successful hit against a “living” enemy will cause it to take damage. |

## Environment

|  |  |
| --- | --- |
| **Object** | **Details** |
| Standard Platform | Immovable.  Unclimbable.  Can be stood on unless hazardous. |
| Climbable Platform | Immovable.  Can be climbed by players.  Can be stood on. |
| Movable Platform | Movable.  Can be stood on.  May be able to be climbed on (think ladder).  May require both players grabbing it to push/pull object. |
| Levers | Grabbable.  When used:  - Move one or both player’s string anchor points to another position.  - Move a set piece from one place to another.  Sometimes necessary to solving levels. |
| (Bonus) Props | Objects that can be held in hand.  Can be used to fight against enemies (hold and press attack). |
| Audience Shadow | Cannot be directly interacted with. Will appear to silhouette players when they take too long to complete an objective. Shortly after a tomato will be thrown which will kill the player if it makes contact. |
| Player Shader | If a player goes behind stage objects, a wireframe view should allow the users to see where they are. |

## Strings

|  |  |
| --- | --- |
| **Ability** | **Details** |
| Anchoring | Each player is anchored by a string from their centre of mass to an anchor point on the top edge of the screen.  Some of the string may lay loose on the floor.  If the player runs to the maximum extent of the string, they will be unable to move any further. (Bonus) a struggling animation will play demonstrating that the player is at the end of the string. |
| Tangling | By walking around one another’s anchor points the players can wrap their strings around each other’s strings and poles in the environment. This can shorten the string length and allow them to complete puzzles that would otherwise be inaccessible. |
| Climb | Players can climb each other’s strings to reach inaccessible places. They can also climb their own strings. This is useful for swinging (see below). |
| Swing | Players can swing on their strings. To do this they must be holding the string and not touching the ground. This should function with standard momentum meaning it will take multiple swings to reach maximum swing height. The players will not be able to swing themselves past 60 degrees (subject to tuning). |
| (Bonus) Slingshot | After an upgrade to the default rope-based strings, the strings will be able to stretch. The stretched string will change colour based on how stressed it is and if released when stretched, will throw the player in the corresponding direction defined by physics.  A player can run backwards and stretch their own string but only halfway. If they jump at that point, they will be thrown back a little bit, but this will be mostly impractical for solving puzzles.  To reach full slingshot capacity, player1 must grab player2 and they must both walk back together. Then player1 can release and jump at the same time to achieve maximum slingshot power. |
| (Bonus)  Grapple | Late game, after story progression, players will be able to pick up their anchor points and use them as grappling hooks.  By aiming with direction input, the player can then press the attack button to throw their anchor point. If this hits a viable platform object, it will stick to it, functioning as normal but from the new position. A “recall” button will be used to rip the anchor point out of a position. It will then fall to the floor. The player can retrieve it by grabbing their string and using a direction input to pull the anchor towards them until it is in their hand.  (Bonus) the grapple could be used to damage enemies if thrown correctly. |

## UI

|  |  |
| --- | --- |
| **Display Object** | **Details** |
| Objective Display | If used, will display a text description of the current objective: e.g., “Get both players to the other side of the water.” |

## (Bonus) NPCs

|  |  |
| --- | --- |
| **NPC** | **Details** |
| Stagehand | Modelled by a shadowy dismembered hand.  Will take two hits with any weapon to kill.  Kills the player on contact.  Can be aggressive or non-aggressive. If aggressive, will leap along the ground or in an ariel arc towards players.  All movement and physics will function the same as the player, however stagehands will be faster on the ground (subject to tuning). |

# Architecture

## Overview

We will use a single scene for the bulk of the game.

The levels will be deployed as prefabs and hidden/unhidden as necessary.

## List of Classes

|  |  |
| --- | --- |
| **Class** | **Details** |
| Game Manager | Responsible for managing main game loop code. Will control things like when the UI is shown as well as saving/loading. |
| Level Manager | Responsible for hiding/unhiding the environmental objects. |
| (Bonus) Spawn Manager | Responsible for instantiating new NPC’s if we end up implementing NPC’s. Also responsible for respawning key items if players accidentally loose them in an environmental hazard. |
| **‘Living’ Object** | Parent class of all game objects which can move around the scene exhibiting the traits of being alive. May provide some modularity for shared abilities such as moving between the player class and the enemies. |
| Puppet (Player Class) | The player class. Will importantly contain object reference to the other player and vice versa. This is important as their relative positions will need to be referenced to enable string tangling etc. |
| Enemy | Parent subclass of anything that can hurt the player which can also move around like a player. May contain basic AI features such as reference to the players and aggro distance etc. |
| Stagehands | Type of enemy. |
| **Interactable Object** | Parent class for a subset of interactable objects that can do something in the environment. |
| Lever | Child class of interactable object which will do something when pulled. |
| Button | Child class of interactable object which will do something when pushed. |
| Lock | Chile class of interactable object which will do something when interacted with via key. |
| **Grabbable Object** | Parent class of different types of props that will be able to be picked up and carried. |
| Weapon | A game object that can be held by the puppets and used to attack. |
| Key | A game object that opens locks. (Bonus: may use an interface to kind of hack multiple inheritance and make any props inherit the traits of **interactable object** as well if this turns out to be useful) |

## Inheritance Diagram

Graphical user interface, diagram

Description automatically generated