Alex Adamson aadamson@stanford.edu
Aaron Brown abrown94@stanford.edu
Thaminda Edirisooriya tediris@stanford.edu
Andy Lamb andrew.lamb@stanford.edu
Morgan Tenney mtenney@stanford.edu

Overview

Description of the Project

We are building a two-dimensional platform game. We have already designed a simple level and decided on a basic set of guidelines for our game.

- *Two-dimensional*: Because we have limited time, we are focusing on making the gameplay interesting. Making a three-dimensional game with graphics would force us to spend the majority of our time learning and building 3D models and textures for the game, instead of coding interesting mechanics.
- *Platforming*: We discussed whether to do a “top-down” or “side-scrolling” game. Pacman is an example of a top-down game. This style game potentially yields larger and more varied worlds. Mario is an example of a side-scrolling game. Many side-scrolling games involve jumping between platforms with a gravity component - we call this broad class of games “platform” games. We decided the physics offered by this style of game created a lot of opportunities for fun gameplay. Moreover, a side-scrolling world can be expanded in the vertical dimension (above and below the player) and beyond the left and right edges of the main camera, so a side-scroller allows us to build expansive worlds with interesting physics.
- *Puzzles*: Our game will require thinking and problem solving, instead of being purely about reaction time or button-mashing. Each level will involve players being spawned at a start state (or possibly different start states) and then avoiding traps to reach a finish state. Avoiding traps will involve using a set of tools that allow you to interact with the environment in different ways (i.e. grappling hooks, bombs, lanterns, traps). Different tools will be given to the player for different puzzles, offering multiple interesting ways to solve a single puzzle.
- *Cooperation*: Our game will be multiplayer, and will have a strong emphasis on cooperation between players. Every level will require both players reaching a finish or accomplishing a goal in order for the level to be passed. No level will be possible without the players working together, i.e. it will never be possible for the players to ignore each other and reach the finish. Although we are not planning

out the entire game initially, we have decided on a core set of gameplay components to facilitate and encourage collaboration. The puzzles will be solved by a set of tools; for example, players might need a grappling hook to cross a pit of spikes or a torch to illuminate a dark room. Each player will only be able to carry one tool at a time, so in order to use tools simultaneously, players will need to collaborate. The players will also be able to throw each other further than they will jump, so there might be a pit that players can only cross by throwing each other.

- *Procedurally generated*: Levels will be randomly generated in each time they are played. For example, the distance across a pit or heights of both sides might be change, but it in a way so it will always be possible to complete the level based on the physics of the game. Players might be able to provide a random seed for the level, such as a word or song. Non-gameplay elements, such as art and architectural elements, can also be procedurally generated. Popular examples of this kind of technology are Minecraft and No Man's Sky, both of which use variants of 3D Perlin Noise to generate large, varied landscapes.

Need for the Product

- The video game industry is based on giving consumers a chance to escape the the daily humdrum of normal life through fun and engaging gameplay, and our video game will be no different. We hope that our game will appealing to players of both genders and of multiple age groups. We also will provide some interesting collaborative puzzles for friends and family to solve and bond over through the medium of a platformer.

Potential Audience

- Procedurally generated, cooperative side-scrollers have a proven audience. Similar titles by small studios, such as the survival and crafting game *Starbound*, have received strong sales in spite of modest marketing efforts. *Terraria*, a survival game with a similar art style released in 2011, still controls a spot on Steam's monthly top ten best-selling list, putting its monthly sales nearly five years after launch above ten thousand copies. *Starbound* (another similar title by a small studio) has similar sales figures two years after public release. Moreover, 2D puzzle and survival games have in general witnessed a mass market renaissance with titles across a number of platforms (including PC, mobile and recently traditional console platforms) and consumers have demonstrated a willingness to pay a modest price (around $10 or $15 for the titles mentioned above) for them.

- The audience we intend to cater to consists primarily of well-educated young adults. We hope to build a game that is cerebral and requires significant, non-formulaic thinking to address the challenges presented within it. The multiplayer aspect of the game necessitates the player be in social circles wherein a number of members are interested in this sort of challenge, and our intended audience fits that mold. Moreover, the success of the LAN and hosted multiplayer scene for *Minecraft* demonstrates the existence of such an audience and its willingness to engage in games that present challenges that require cooperation.
- We also want our game to have mass market appeal beyond the typical "gamer" demographic. In that vein, we intend to not make the game be unduly counterintuitive to play or advance within. The puzzles within the game should seem clever and fulfilling in hindsight rather than cruel or needing a "trick" to solve (as is the case in some 2D platforming "mindf**k" games such as *Pony Island*).

Discussion of Competing Products

- The obvious competing products are 2D games that feature similar thematic or gameplay elements (such as Roguelikes and puzzle platformers).
- **Spelunky** - A 2D Platformer "Roguelike" game, featuring partially random levels, exploration, action, and permanent death. Our game seeks to open up parts of this experience to multiple players over a network connection, something Spelunky, and most other games of the genre, fail to do.
- **Monaco: What's Yours is Mine** - A top-down 2D multiplayer heist game that pits players against AI guards and each other to steal as much money as possible from the banks you rob, and escape. While it captures much of the same multiplayer feel and camaraderie, the game has no meaningful physics, something that we intend to feature as a major component of our puzzles.
- **Super Meat Boy** - 2D platformer that featured a fast pace of gameplay in short levels, known for being incredibly punishing. Again, no multiplayer support is where the product differs from ours, as well as fixed levels with no opportunity for new environments and challenges.
- **No Man's Sky** - 3D first person exploration game that takes place in space and on planets inhabited by dinosaur-like creatures and by robots, as well as other players. While the game is multiplayer, direct player-to-player interaction is not necessary at all in order to play the game, something that we are striving for with our game. We also forgo 3D graphics in favor of simpler 2D graphics that allow us to put more focus on individual gameplay mechanics and the network infrastructure, rather than on visuals.

Major Technologies Used

- We will use the Unity game engine to develop the actual game. Games are created on Unity using a GUI and C# scripts. One nice aspect of Unity is its ability to target multiple platforms - we will primarily develop for OS X, but should be able to port to iOS, Android, Windows, etc. without requiring vastly separate code bases. A majority of our team members have used Unity for video game projects.
- For artwork and sprite creation, we will use Aseprite and Photoshop. We plan to use freely available sprite packs as starting points for the artwork.

Resource Requirements

- The community version of Unity is free, so we should not need to spend money on major resources such as licenses. In order to distribute the game, we would need to get a distributor's account on Steam Greenlight and the Apple Developer Program, both of which require significant membership fees.

Potential Approaches

- Instead of using Unity to develop our game, we could have used one of a myriad of other Game Engine technologies - Unreal Engine, Cocos 2D, LibGDX CryEngine
- We could also have written our own game engine based on OpenGL, WebGL, or DirectX, giving us more fine tuned control over graphics, effects, memory management, etc.
- Using Unity, though, eliminates the large chunk of time that would otherwise have been spent learning graphics programming, and allows us to spend this time developing our game mechanics, visuals, and systems to make it as fun as possible, without getting bogged down solving problems that have already been solved far better than we could in 10 weeks.
- We could manually generate our levels for our game. However, this would result in more time spent tweaking individual levels, and less replayability in the final product. By using procedural generation, the game will be more interesting to play on subsequent attempts.

Assessment of Risks

- Being a networked game, having the game feel smooth on every client while also simulating accurate physics for each player is going to be a challenge. In particular, inter-character interactions such as picking up other players will be tricky to handle in high-latency situations. We plan to develop this game as

primarily a Local Area Network game to help mitigate this, but nonetheless it will be difficult.
- Group dynamics
- Generating puzzles with a procedural content system will be tricky, since balancing diversity of puzzles and platforms with playability is not easy to do automatically and without testing. Procedural generation as a technique is not fully explored even in the professional game development community, and many games have struggled to make interesting content that feels novel each time.
- Story: we have yet to consider whether or not story should be a strong aspect of the game, and none of us are particularly good narrative writers. A lot of games feature a strong narrative as a motive to keep players playing, and if our game lacks the ability to keep players coming back, story elements are something that we may need to consider.
- Some of us have never developed a game before/haven't used Unity. Game Development in general is also a time consuming endeavor since it necessitates a large amount of code and testing, as well as art, design, etc. Completing the fully featured game within the time constraints is not an easy task.

Next Steps

- We have already created a simple networked client for our game that includes basic mechanics (such as jumping, grabbing and throwing other players).
- We have mocked up a simple puzzle that needs only two players and jumping/throwing mechanics to solve. We need to implement this level in Unity and then play it while fine-tuning elements of the physics, player movement, and network interaction. Once the in-game experience (i.e. feedback from controls seems smooth and not surprising) is satisfactory and stable, we can begin experimenting with artwork and the ambience of the game while being confident that the experience we're crafting will be consistent with the experience of the final product (since the interface controls and their feedback has been pinned down).
- Before we can begin creating an engine to procedurally generate rooms, we need to manually construct what we think are suitably challenging puzzles and analyze why they are challenging and how we went about creating them. Over the next few days, each team member is responsible for doodling, analyzing, and explaining a room. Our hope is that the process of thinking through how to create a solvable but challenging puzzle will enable us to write down a set of principles and procedures that we can then use to create the world creation engine.
- Our prototype multiplayer game works, but movement on the client still needs to be tuned to feel better, more like a mario game. We will spend some time just

tweaking the player movement in a simple world to make sure that running, jumping, throwing, etc. all “feel” fun before we get too far into developing other mechanics.