Sam Levatich

Procedurally Generated Landscapes For Sandbox Play
Advised by: Holly Rushmeier

Abstract:

Game development, as a large commercial realm with lots of worlds to be generated, stands to benefit a lot from a smart procedure for generating different sorts of levels or landscapes to play in. But the single most important quality of a game is a simple question: is it fun? The answer to that question is, unfortunately, subjective and complicated, but principles of game design can give some answers. In this project, I created “The Adventures of Moji in the Land of Semi-Random Quilts,” a cheekily titled proof of concept game in the Unity game engine with a variety of procedurally generated and placed elements that contribute toward player enjoyment. The game puts players in control of Moji, who interacts with the generated landscape through running, jumping, and grappling upward, and aims to collect all the rewards strewn throughout the game world. Runtime generated challenges such as climbing an ever taller series of columns, hopping from platform to platform over dangerous lava, and hunting for rewards within player created geometry provide a sense of danger, accomplishment, and fun, while indirect modes of enjoyment like character emotions and a full fledged selfie mode provide alternate ways to have a pleasant gaming experience. Because it is procedurally generated based on an initial map drawn by the user, no playthrough is the same and it offers high replayability even in proof of concept state. Player enjoyment was measured through qualitative user feedback, gathered during testing at multiple points during the development process.

Game Description:

The player controls a character named Moji and attempts to collect all of the rewards spread throughout the generated map. The player can tune several parameters of the game world, including horizontal and vertical dimensions, density of elements, terrain variance, and which types of challenges (which I refer to as the game ‘elements’) will appear in the world for them to conquer. The player also gets to define the basic terrain layout by drawing where on the map they would like higher or lower ground (which take the forms of mountains and drained lakes). To accomplish the goal of acquiring all the spheres the player has the ability, as Moji, to run, jump, and vertically grapple. The primary challenge the player in confronting is how to move Moji vertically through use of terrain elements and their jump and grapple mechanics, and this is reflected in the placement of rewards at different elevations. In order to cater to players less interested in the challenge of collecting spheres, there is also a selfie camera mode which allows for the capturing of screenshots of Moji emoting in various ways (this ability to emote is what gives them their name).

What Unity does and what I do:

Unity takes care of rendering and physics primarily. This allowed me to focus on the procedural generation of mesh geometry and fine tuning of character mechanics. The procedural generation and placement of elements described below is the bulk of my codebase (startup.js). Mechanical control, inspired by unity standard third person controller assets but modified extensively based on tester feedback and my mechanical needs are another key portion (TPUser.js and TPChar.js). Additional work is done inside the editor to correctly set up non random elements of the geometry (the drawing canvas, character colliders, death triggers, all prefab defaults). Camera control is the only piece of code that still utilizes some of unity's standard assets package (hence why it is the only script in C#), but I modified that script to do some autonomous camera movement (again, based on user feedback) and adding in the selfie camera mode. This isn't code at all but all of the animations, models, materials, and textures in the game are of my own creation as well. Sound effects were taking from freesounds.org

The Game Elements:

I identified several metrics which contribute to a fun game experience through my experience playing a lot of games, through play tester feedback, and through trial and error in the design process. Each involves random placement, responds to the way the user chooses to draw the map, and is generated at runtime.

Platform Challenge:

This is the primary, and most traditional way of creating fun in this game world: through a challenge. It's also the portion of the game players consistently liked the most, regardless of self reported comfort with video games. After reserving a 5x5 grid space in the ground portions of the game world, columns are randomly placed in a sequential manner such that one column is always reachable from the other. I am then able to tune various parameters with what I style difficulty functions, which control how parameters like width, height, and distance change over the course of the platform challenge. There are a couple different difficulty functions I tried out, but the one that gave the best flow state (a concept in game design that refers to balancing players' increasing skill level with the difficulty of the challenge: basically making it not too hard or too easy while also growing in difficulty along with the players skill) has been made the default. The grid placement changes every playthrough, as does the arrangement of columns, but the challenge level remains consistent. The columns themselves are also procedurally generated meshes and so no column of the same height and width is the same either. The small and large scale randomness involved in generation of this challenge make this the most successful challenge in terms of feeling organic within the game world.

Hidden rewards:

One of the ways games can delight is exploration, which emphasizes the joy of discovery. Since players are creating their own map through drawing user input, one of the ways I can create a fun game experience for them is to place rewards in such a way that they are encouraged (via the payoff of collecting rewards) to poke around the world they've created, and not just go toward the first more obvious challenge they see. I do this by placing rewards at a low frequency in places the user cannot initially see: this is done by casting rays out from their initial position and if geometry is hit a reward is sometimes placed at the lowest point in the geometry beyond that hit portion of the mesh.

Bouncies:

A very simple way that I created fun is through the random placement of bouncy spheres. Since the character has a limited jump height, to reach an element of the game that launches them far far higher, giving them a view of the world they've created, is incredibly satisfying and often provoked audible oohs and smiles from playtesters. Rewards are placed above the center of the bouncies, which give a sharp impulse upward to launch the player, so players are encouraged to jump on the geometry and thus discover that it launches you upward, at which point you can collect additional rewards placed high above the bouncy. Bouncies also play into the games emergent gameplay: every so often a bouncy is placed close enough to a platform challenge that players can skip some early jumps, which is rewarding the player for figuring out how they can use these two separate elements together to proceed vertically upward.

Emotions and Selfie Mode:

I gave the character model a variety of emotions because I wanted Moji to be a likable character and animation is one of the ways users can bond with a character. But then I ran into an issue, which is that the character is often facing forward so the emotions are unreadable. This lead to the establishment of selfie mode, which moves the camera in front of Moji and zooms in, to allow you to take close up shots. This doesn't contribute to the goal of spheres but rather offers a different way to play: it's much more social since the screenshots can later be shared. This also plays into the joy of exploring a world you've created mentioned above: taking pictures at high points with triumphant emotions reinforces achievement of objectives and if you fall and fail to collect a reward, triggering the sadness emotion can serve as a break from the action for a moment of empathy.

Platform Waveys and Danger:

Falling is a burden in the column challenge, and players found it punishment enough. But I wanted to also create horizontal challenges to give the player an opportunity to learn how to best fine tune horizontal jumps, but falling in a horizontal sense doesn't punish the player (since you can just run back to where you were). By putting a set of platforms (and of course, rewards) over what I call a wavey, which is undulating in real time and uses a lava texture, the player is given sufficient motivation and challenge to practice this other crucial skill. The platforms are the same material as in the platform challenge, reinforcing both as similar challenges.

Overhangs:

Grapple offers a different mechanic for rising that is different from jumping, and this is a challenge that allows for using it. This is mainly in the game to exploit the option to have geometry above your head which creates a more visually exciting world, but similarly to the bouncies, you can jump from overhangs over to platforms or other elements which give a sense of fullness to the world. By pressing the 'Y' button on a controller or the 'g' key on a keyboard, Moji will grapple up to a piece of geometry above their head for a certain distance and if it connects will arrive on the top of the geometry at the conclusion of the animation, where a reward awaits.

Other Smaller Contributions:

* Small amounts of randomness and a unified aesthetic are key to making the game less clinical and more organic.
* Visual delight is key to creating a world fun to play it: thus the game is colorful and covered in a quilt texture to make an inviting landscape to run around it
* Editing mechanics to feel natural was the main result of extensive playtesting. This includes how fast the character ramps up movement, how the camera moves autonomously to assist players, and different physics acceleration values (like gravity).