Metroid Clone: The Game

Tanner Evins

November 13, 2015

1 Hello!

If you're reading this, you're one of the fortunate souls chosen to help create the next cutting edge video game (or something close). We're lucky to have a varied group of talented folks. If a bunch of hobbyist writers, artists, and programmers—all of whom game themselves—can't make a game of their own, I'm not sure who can.

Our first project is a relatively simple one: 2D Metroidvania-style platformer. Yes, this game will be wonderfully derivative, but that's okay; it's intentionally so. To my knowledge, none of us have made a game before, so it would behoove us to limit the variables of our project as much as we can. Making a game where we don't have to worry about innovating gameplay mechanics will allow us to become comfortable in our duties as well as working with each other. After this, we can move onto more ambitious projects.

Let's make a game, y'all!

A note about the document. You'll notice that this document is a PDF rather than in a more collaborative format, like a Google doc. The reason for this is that I believe it's important for there to be one director for the game and one document to govern the requirements for the project. Individuals are of course free to collaborate among themselves how they see fit, but the purpose of this document is to capture the spirit of the finished game as well as to enumerate the general goals to be accomplished. It is, however, a living document, and will be updated as our requirements become more concrete.

2 Metroid Review

The game is a Metroid clone. We'll have a lot of creative freedom as far as world-building, dialogue, and artwork goes, but the *mechanics* of the game are straight from Metroid.

Elements

There's no better exposition on the mechanics of (Super) Metroid play than the game itself—which you can do fairly easily with emulation or the Wii U's virtual console—but I'll attempt to enumerate the key elements here.

Exploration. Metroid games all begin with Samus landing on a foreign world. She's left to her own devices in a strange environment. It's up to you, the player, to discover and explore the world and keep a mental map of it as you go. More on this later.

Item Discovery. Along the way, the player encounters items that upgrade Samus's abilities. She doesn't begin with the morph ball ability, for example—she acquires it. Having to discover items instead of starting with them in your arsenal ties into the exploration aspect rather nicely. More often than not, a new item will allow you to access previously unavailable areas. The morph ball enables Samus traverse tiny tunnels.

Now imagine if the player began the game with the morph ball. The player would be free to travel these tunnels as soon as they're encountered. Delaying the acquisition of the morph ball forces the player to mentally mark those tunnels as significant for a later time. This enforces the player's habit of keeping track of what the world looks like while playing. Very satisfying.

Nonlinearity. The original NES Metroid begins famously with the player having to go, not right, but left to progress through the game. This broke the precedent established by platformers like Mario where the player continues right until completing the level. In Metroid, the player goes left, right, up, and down; no direction is privileged as the "right" way to go (pun intended).

Two Levels of Play

I've implied this in the section above, but I'll state it explicitly here: *The essence of Metroid gameplay lies in its two simultaneous levels of play: (1) The immediate situation that demands that you respond to the dangers and puzzles in the room/area, and (2) the high-level tracking of where you are in the world and where you need to go next.* These two levels work together to sustain the players interest to complete the game.

If you take any given room in Metroid, there's only so many things that can hold your interest. Yes, there will be a few enemies that block your way, but you've likely encountered their type before and know how to deal with them. If you haven't, then you deal with them once and can handle them any time in the future. If a Metroid game were to consist in a series of superficially connected rooms where the player overcame repetitive situation after repetitive situation, then the game would suffer for it.

This is not the way Metroid games work though. The rooms are connected much more deeply. I mentioned above that having the player *discover* items forces the player to make a mental map of how the world appears as a whole.

See an odd looking wall? Maybe it's breakable. I'll remember that. Oh, I've just discovered missiles! I should go back and see if I can destroy it. Praise the sun, I can! Now I'm off to a new area. And since I wasn't able to do that before, I feel that I'm progressing.

It is this kind of gameplay that gives Metroid games their enduring appeal. Combat gameplay has unquestionably improved since the days of the SNES, but players today can still appreciate the high-level thinking required to progress through a Metroid game. It bestows on the world a holistic quality not found in its peers. A lot of good games today take their cue from Metroid in this respect. (Dark Souls and Bloodborne come to mind in particular.)

For a more in-depth overview of this aspect of Metroid, I highly recommend reading "The Invisible Hand of Super Metroid" on Gamasutra.[1]

3 Responsibilities

A finished game is rarely the result of a singular mind—unless that mind is capable of drawing, sound production, writing, programming, and level design and also has the time and resources to give each of these components its due. Obviously, you'd be hard pressed to find an individual that meets that criteria.

Games are made by teams. The final game may have spawned from the vision of one person, but its realization comes from the minds of several people. It is my hope that each of you will be responsible for a well-defined domain of the game's creation. You may work together—in fact I encourage you to—but at least one concrete aspect of the game will be your responsibility. Let's attempt to enumerate these roles and responsibilities here.

Directing

The role of the director is to manage the other components of the process to ensure that the final game is a consistent whole.

Writing

Preliminary—especially the last two distinctions. Writing is not my craft. Divide the responsibilities in a way such that I can hold each of you accountable.

World Building.

Plot.

[1] `http://www.gamasutra.com/blogs/HugoBille/20120114/90903/The_Invisible_Hand_of_Super_Metroid.php`

Characters.

Art and Visuals

Sound Design

Level Design

Scripting

Programming

Testing

4 Tools

Github

Github is a collaboration website for storing code repositories. It's an excellent platform for team members to track issues and project milestones. Its use does not require knowledge of the Git version control system, though some know-how in it can go a long way. As a team, we will be using it primarily as an issue tracker and as a hub for team communication.

You'll need to create an account in order to contribute. The repository home is currently at `https://github.com/evinstk/MetroidClone`. Notify me of your account name and I'll mark you as a collaborator. From there, you can click on the "Issues" tab in the right column. This is a user (evinstk—me) repository; we may want to switch to an organization repository if we want to be stricter on modification control, but for now, we'll keep it simple.

Tiled

Tiled is an open-source map editor that creates data files to be consumed by game engines. It works by loading image files (e.g, JPEG, PNG, etc.) as tilesets—that is, an image divided evenly into cells/tiles of a certain width and height. The user arranges the tiles into a map of their liking, then exports their work into a data file to be read by an engine. Tantech engine currently supports the Lua data format.

Tiled also contains a basic animation editor. The user can specify a sequence of loaded tiles and assign a duration for each one to be displayed. The effect may be a character walking animation, tree leaves rustling in the wind, or anything that can be represented by a series of two-dimensional sprites.

Finally, Tiled can also define collision geometry. Say you have a tile of a boulder. The user can specify an area on the boulder that an avatar would collide with. Currently Tantech engine only supports rectangle geometry, but this is sufficient for most needs.

Level designers will use Tiled as their primary tool. Artists may also wish to use it for its simple animation editor, though there is likely a better tool for that purpose.[2] You can download Tiled at `http://www.mapeditor.com`.

Git (for the brave)

Git is not the same thing as Github. Github is a collaboration website that hosts Git repositories. Git is a version control tool; it records the change history of a project through manipulation of "branches." I will be using Git to store engine code as well as to integrate your changes.

I will not require you to learn Git. If you're new to version control systems, the nuances of Git can be especially tricky to grasp conceptually. That said, if you're willing to take the time to learn the tool, it can dramatically improve your workflow on collaborative projects like this (and not to mention take a significant load off me integrating your changes). You'll need to install the tool itself at `https://git-scm.com`. The best resource to learn Git is the online book Pro Git at `https://git-scm.com/book/en/v2/`.

[2]Tantech only supports animations defined in Tiled currently. If an artist intends to use a different tool, inform the programmers as soon as possible so they may support it.