Research and Analysis

EECS 494 W17 Team 2

Justin Fan, Cameron Gagnon, Stephen Kline, Alex Wang

Oxygen Not Included - Klei Entertainment - PC

Oxygen Not Included (ONI) is an underground colony simulation game recently released onto the Steam platform. Despite being in an early access state, the game already features a polished building system and thematic flair usually found in finished products. The player is an omniscient third-party in charge of delegating commands to a small crew of characters around the viewable colony by selection and hotkeys with the mouse and keyboard. Although this control scheme differs from our game's focus on directly controlling the player character, ONI's manipulation of its 2D environment provides many valuable takeaways for our development.

For starters, every inanimate object or material in ONI is located and created on a grid system reminiscent of a Duplo-scale version of Terraria. This breaks the game world up into visible chunks and forces the player to interact with it in discrete units. Although this approach can stifle creative flexibility, it also makes it much easier to create visually pleasing colony layouts and construction projects, as well as decreasing the overall complexity of the game.

Construction in ONI comes in two distinct flavors. There are many pre-designed small buildings available that can be placed anywhere, provided that the raw materials are available and the player's minions can safely reach the location. These buildings all serve different gameplay purposes, but cannot be built on top of or directly expanded. The second way is the ability to build up the colony itself using different kinds of tiles and doors to decide the border of the player's base. In addition, many of the aforementioned special structures must also be connected to resources or other structures by building pipes and wiring into the grid system. Structural elements such as tiles, doors, and pipes, appear to be supported by the background and are not susceptible to gravity, although raw materials, fluids, and sand blocks can freely fall from one grid block to another. In addition, blocks can be deconstructed by player command, or even destroyed by extreme fluid pressure or temperature.

Although a two-dimensional game, ONI packs a large amount of complexity and detail into every square of its building grid. Ground, tiles and doors will block character movement as expected, but unsightly pipes and wiring do not and can actually be built into tiled blocks to improve “decor” (a valuable parameter for stress reduction in every underground cavern base). Special buildings also are built into the background and do not restrict movement while also allowing characters to interact with them. In addition, every square has temperature, air composition, thermal conductivity, and many other constantly-changing statistics. ONI is a great example of the depth, pun intended, possible when interacting with a sub-3D environment.

Lastly, I greatly enjoyed the amount of personality already present in ONI. Character faces are unusually large and feature melodramatic responses to game-critical effects such as asphyxiation, high stress, sleep, and even bathroom cleaning duties. Character and building animations and motion are exaggerated to both convey functionality and crack smiles. Sound effects are abundant, and the scales and arpeggios that play when selecting large swaths of grid with the mouse are gloriously satisfying touches. The game’s cartoon style may not be appealing to everyone, but it gives me a juicy vision to strive for in my own 2D games such as our final project for the course.

World of Goo - 2D boy - Mobile devices (iOS & Android)

World of Goo is a tower building game that lets the player build with goo balls in order to build around obstacles and reach the objective (a pipe). Once this objective is complete, the remaining goo balls are sucked up into the pipe. The target amount of goo balls must be collected in order to proceed to the next level. The interesting part of this game that I will be analyzing is the building mechanic of it.

The building mechanic is important as it highlights a few things that create interesting gameplay. First is the way in which you build. It is not like Tetris, where the blocks fall and you move them around; instead, you pick up goo balls and move them into position where you would like to build. This mechanic is nice, since you can have total control over where you place the goo balls and there is no grid system to form perfect and neat towers. Oftentimes, the challenge is trying to build a structurally sound tower by yourself, which is a large aspect of the game. Although this lack of grid system fits for this game, there will likely be a grid system in ours. Despite having a grid system, there will still be elements from World of Goo present, such as building towers that don't quite make physical sense. This will allow for interesting aspects such as building out on an edge, and building towers that may be wider on top than they are on bottom. This adds strategic complexity to the game through basic building mechanics.

Something that is required with having total control over positioning is knowing where you are allowed to build. A very nice mechanic in World of Goo is that outlines of lines will light up when a goo ball is in a position it can be placed. This lets the player know exactly what positionings are allowed, and what are not. That way they won't get frustrated when they repeatedly try to place goo balls that are in invalid positions. This aspect guides the player into a smooth building experience, as it becomes fairly obvious pretty quickly as to how to actually play the game.

Thanks to simple building rules (that allow for complex strategy and free range on what towers can be built), and a good guiding aspect (that highlight the locations a player can build), World of Goo has a nice build mechanic that allows players to jump in quickly and not get frustrated in the building process.

Bowmaster Prelude - Lost Vectors - PC (lostvectors.com/prelude/)

This game allows the player to shoot different types of arrows at the opposing castles. The enemies come in waves and you must destroy them before they take your flag. The part I'd like to look closer at is the attack system of the game.

Initially, all you have is a bow, with a cooldown time between shots. This aspect of the game makes the attack system more complex by forcing the user either to wait between shots, or to switch to a different bow and use that one while the previous bow cools down. Once you level up and unlock more bows, you can have up to 9 different bows. At that point in the game, you can almost always be shooting something because each bow has a different cooldown time and you can switch back and forth between many bows. This provides a limiting factor to each individual bow, but when combined as a whole, makes for a complex and strategic attack process. By knowing how fast the cooldown times are, and about how much damage each one does, the player can shoot in specific orders and use weapons at specific times to deal the maximum damage when needed.

Having a cooldown period instead of mana or ammo allows the game to achieve a similar effect of mana, which is that of limiting the amount of times a player can shoot. Instead of adding another “resource” to the game that the player has to think about, all they have to do is focus on how much time is left until they can use that weapon again. This seems more elegant than adding an ammo system to Bowmaster Prelude, especially because there are 9 different bows the player would have to think about, and it could get more difficult to cleanly identify how much ammo each bow has left in a round.

Utilizing a cooldown phase allows for fewer numbers the player has to keep track of, a cleaner ammunition experience, and when added together with other bows, a uniform “when-can-I-shoot” experience.

Minecraft - Mojang - Originally for PC (2011)

There is a lot to say when discussing *Minecraft*. The famous 3D block building sandbox game is known for its quick rise to fame and seemingly unavoidable presence on nearly every game console and entertainment medium. Yet, no matter what field *Minecraft* is involved in, education, virtual reality, augmented reality, etc., at its core, the game revolves around the spirit of providing building blocks for players to construct anything they can to achieve their goal.

Thrown into any kind of environment (excluding creative mode which provides infinite resources), players have the freedom to collect and spend resources however they see fit to complete their self-defined goal. This scarcity provides a very organic type of puzzle for the player. Players can only gain from collecting and crafting. Thus, the only and greatest cost to consuming resources is purely opportunity cost. For example, players can use their freshly harvested wood to create a weapon to defend themselves, a tool to harvest more wood, or various forms of shelter. Materials are flexible, time is limited, and opposition is often harsh. The challenge of the game comes from whatever reason a player has to collect and craft in the first place. Players define their own win criteria and the design of the game gives players plenty of opportunity to explore and experiment to make the best decisions to achieve said goal.

This decision making is especially important when competing with other players. Since using up resources leads primarily to positive results, the key to winning no longer becomes simply using enough resources, but transforms to using resources better than everyone else. *Minecraft's* natural survival mode, and many other custom multiplayer modes place all players on equal footing where winning players are decided by whoever can pull ahead through good decision making. Sometimes this doesn't require any careful planning, but just awareness of what opponents are doing. Players who can adapt well to what they have and what others are doing are naturally rewarded with more resources. While this does create a "snowballing" effects for those who are winning, the game's just mechanics just as easily punish players for poor decisions. There is no purely correct or incorrect choice, and *Minecraft*'s wide array of materials, crafting recipes, and hazards provide an experience that fosters not only experimentation, but also creativity in tackling whatever challenges the cruel *Minecraft* world throws at the player.

Kerbal Space Program - Squad - PC

Kerbal Space Program is best described as an aerospace sandbox game. In the standard game mode, the player is not given an official goal, only the tools to build and pilot his/her own spaceship. The physics in the game are heralded as being incredibly accurate, with even small details being taken into account like the change in weight of fuel as it is burned up. The game itself is less about challenging the player as it is a playground in which the player can challenge themselves. There’s nothing saying that the player should try to land a shuttle on the closest space object (the Mun in this case), or that they shouldn’t just strap fifteen solid rocket boosters onto a command module and light it off. The fun of this game comes from just playing around with the tools given to you, which in this case is a whole bunch of rockets and things to attach to those rockets.

The rocket building system in this game is what is most interesting when compared with our game. In recent updated versions of the Kerbal Space Program, there is now a more fleshed-out tutorial to guide the player through the basics of building their first spaceship, but it wasn’t always that way. A big part of Kerbal Space Program that I enjoyed was simply figuring out what my limits were in terms of what kinds of physics interactions were actually taken into account by the game. So sure I could strap eight solid rocket boosters to each other in the hangar and everything looked fine, but once I tried to launch my monstrosity it would become very apparent that adding structural components would be just as important as the number of rockets I used. Similar to this system, we want our game to show the player how to use the tools they are given, but not guide them in terms of what they can build. In too many games I have seen the tutorial show the player exactly what they should do, and this often just confines the player into that kind of thinking, instead of exploring all of the ways they could push the boundaries of what they can do.

SourceForts - Half-Life 2 Deathmatch Mod - PC

SourceForts was a multiplayer mod created for Half-Life 2 Deathmatch in 2005. The main gameplay was capture the flag on symmetric maps, but with an added twist. A regular game was split up into two phases: building and fighting. During the building phase, the teams would be separated by an impenetrable wall running through the middle of the map. Players would not have any weapons during this time, so there was no fighting of any kind. Instead, players were armed with the iconic gravity gun from Half Life 2 what they could use to transport, position, and lock in place panels that could be used as anything from a basic barricade wall to an elaborate tunnel system that slowed down attackers trying to reach their flag. Once the building phase was complete, all hell would break loose. Players were then granted weapons that they could use to attack enemy players or target enemy barricades. There are many aspects of this game that we looked to for ideas in terms of how we wanted our game to play.

The building phase of SourceForts was pretty straightforward. Teams were given a time limit and block limit that they could use in whatever way they wanted. Both of these limits were lax enough that some actual planning could be done during this time, so that the team could coordinate what they wanted their base to look like and they could avoid having to frantically run around hacking together a defensive structure. Creating and placing the individual blocks was easy enough, but the real limiting factor on many maps was the distance players would have to travel between the block spawners and where they wanted to put the blocks. Experienced players would work together by having one spawning blocks and launching them to their teammates to cut out the time needed to transport the blocks. This is a feature that we plan on directly adding to our game.

During the battle phase, the teams would play capture the flag against each other, trying to navigate their opponent’s newly built fort. Each block had a health associated with it, but it was great enough that it would take focused direct fire for a substantial amount of time to destroy the block. If a player wanted to take down the other team’s blocks, a much faster but riskier method would be to stand right next to it and “unweld” it. This would not only remove the block from its current position, but also change its color, indicating that it was now the other team’s property. While new blocks could not be spawned during the battle phase, existing blocks that had been unwelded could be placed in new positions. Doing this was much slower than the instant welding that was possible during the build phase. This pushed the focus of the battle phase to actual fighting, something we want to implement into our game as well.