

March 2011

1906 Expressing Disaster in Games

Andrew D. Tremblay
Worcester Polytechnic Institute

Joshua Adam Ginsburg
Worcester Polytechnic Institute

Matthew G. Ivory
Worcester Polytechnic Institute

William E. Early
Worcester Polytechnic Institute

Follow this and additional works at: <https://digitalcommons.wpi.edu/mqp-all>

Repository Citation

Tremblay, A. D., Ginsburg, J. A., Ivory, M. G., & Early, W. E. (2011). *1906 Expressing Disaster in Games*. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/8>

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

1906

Expressing Disaster in Games

A Major Qualifying Project Report:
Submitted to the faculty
of the
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
By

Andrew Tremblay

William Early

Josh Ginsburg

Matt Ivory

Date: March 8, 2011

Approved:

Professor Dean O'Donnell, Advisor

Professor Britton Snyder, Advisor

Abstract

1906 is a serious first-person exploration game designed to increase the salience of a user's own mortality while providing an engaging rendition of historical events. The game is set in San Francisco during the Great Earthquake and Fire Disaster of 1906, one of the most devastating natural disasters of our nation's history. Players will have to interact with virtual citizens as well as try to keep their virtual self alive. Using its unique ambience and setting, 1906 allows players to experience life in the aftermath of one of America's greatest natural disasters.

Table of Contents

Abstract.....	2
Introduction	4
Background.....	5
Methodology	10
Organizational Approach	10
Art Design	11
Tech Design	13
C#.....	15
Singletons.....	16
UniSciTE.....	17
Singleton Dependency Outline.....	18
Organizational Design	19
Development	22
Unity 2.8 to Unity 3.....	22
Ambient Occlusion Culling	22
MonoDevelop.....	23
Third-Party Scripts and Libraries	23
AngryAnt's A* Pathfinding	23
Flickering Fire Script	24
Unity's Procedural Animation Package	24
Art Development.....	25
General Asset Workflow.....	26
Character Workflow	30
Workflow Complications.....	33
Flame Development.....	36
Heads Up Display	37
Organizational Development	39
Conclusion	41
Appendices	42
Appendix A Art Assets	42
.....	42
.....	43
.....	49
Appendix B Script Index.....	50
Appendix C Art Asset List	52
Appendix D Mechanic Description.....	55
Appendix E: Calendar	60
Appendix F: Tech.....	66
Bibliography	67
Appendix G: Story Throughline	68

Appendix G: Task List.....	69
Appendix G: Barks	71
Bibliography	72
Resources.....	Error! Bookmark not defined.

1906: Expressing Disaster in Games

“No one can comprehend the calamity to San Francisco in its entirety. The individual experience can probably give the general public the clearest idea. I was one of the fortunate ones, for neither personal injury nor death visited my household; but what I saw and felt I will try to give to you.”

-Emma M. Burke, survivor of the 1906 San Francisco Earthquake[1]

Introduction

There are many reasons we wanted to make a video game about the 1906 earthquake in San Francisco. It was an event that had catastrophic consequences on the people at the time, and had a financial impact that is equivalent to that of hurricane Katrina's. It has been compared to Haiti in terms of how much devastation was caused. And due to the effects of the people of the city, the situation evolved in a very unique way. What other disaster had people actually setting fire to their own homes in order to get the insurance they wouldn't get from the earthquake?

What we made, however, does not fit into the mold of a typical video game. There are no bright flashing lights, no score, no levels, or anything else that is so commonly associated with the medium. In fact, we are actually hesitant about calling 1906 a game. It does not have a true win condition, and while there are several goals in the game, they are more abstract than the typical game, being to simply not die, and not lose hope in yourself or the city. We wanted an experience that would affect the player and possibly create either an awareness of the earthquake or a desire to look up and learn about the earthquake for themselves. We did not even truly want the game to be fun in the typical sense of the word, feeling that it should be engaging, but that the player should not be playing the game with necessarily a smile on their face. But we are unsure of what else to call 1906 but a game, given the medium it is in.

Our goal was not to simply recreate the earthquake of 1906, but to attempt to condense the experience down for the player. We took several liberties for the sake of the player's experience, but we feel that what makes 1906 unique is the experience it will give the player. We know that we cannot make the player feel the shaking of the earth or the heat of the flames, but we would like to make the player understand the sensation of being trapped in a world you once knew that is crumbling to the ground before your eyes.

Background

On Wednesday April 18th, 1906, at approximately 5:12 a.m., an earthquake struck San Francisco, California. It ruptured along the San Andreas Fault and covered a total area of 296 miles, and had an estimated magnitude of 7.9. And even though over one hundred years have

passed, this event is still the greatest loss of life from a natural disaster in California's entire history. But part of the significance of this event is how the damage was done. Although the earthquake was damaging enough on its own, part of the reason that the damage and effects were so severe was the sweeping fire that lasted for three days, resulting in around 90 percent of the damage.

California as a whole is no stranger to earthquakes. Located on hundreds of active faults, thousands of earthquakes occur per year, although many are so small that they are not even felt. But many are, and the history of earthquakes in California stretches all the way back to 1769, during the first recorded expedition into the area. And so the people who settled and lived in the state tended to have a keen awareness of how shaky the ground they lived on truly was. Buildings, such as the Palace Hotel, were built specifically so that if an earthquake occurred, they would be able to ride out the storm and survive relatively intact. And prior to the earthquake of 1906, the most recent earthquake was in 1899, a 6.7 magnitude earthquake that killed six people and caused over fifty thousand dollars of property damage, an earthquake within a short enough time span that people would not have forgotten the danger an earthquake can have [2]. But even so, the earthquake of 1906 caused much more significant damage and loss of life for two main reasons. The first is due to the scale of the earthquake, which covered an area of over 295 miles long, and with a magnitude estimated to be around 7.8, resulting in a loss of life of over three thousand people. However, what made the earthquake so much more dangerous was not the quake itself, but from the fire that occurred afterwards.

While the earthquake itself was damaging, it was estimated that ninety percent of the damage occurred from the fire that ensued [3]. Over 30 fires raged and burned, destroying approximately twenty five hundred thousand buildings in an area of destruction of four hundred

and ninety city blocks. It is estimated that fifty percent of these buildings were destroyed by firemen who, inexperienced in the use of dynamite, tried to demolish buildings to create firebreaks to attempt to restrict and contain the inferno. Part of this might have been due to the death of the fire chief in the original earthquake itself, but regardless, the fire raged for a total of four days and nights. But even worse were the actions of many towards the fire itself.

Due to the high possibility of an earthquake at any time, many houses were built out of wood and other, more supple materials that were more likely to survive an earthquake than something made out of stone, which would crack and crumble. However, many of these same buildings were not covered by earthquake insurance, possibly because of hubris, or not feeling it would pay off, or many other reasons. But what most buildings did have coverage for was fire insurance. This had two main results. The first was that the majority of destruction caused in the city was attributed to fires, as this would provide the most money for relief. But a second, more unexpected result was that of many people. Due to the fact that their houses were insured against fires, but not earthquakes, and still significantly damaged, many people started setting fire to their own homes. A captain in the U.S. Army Signal Corps helping with the relief effort was “[Stopped] by a fireman who told me that people in that neighborhood were firing their houses, as they were told that they would not get their insurance on buildings damaged by the earthquake unless they were damaged by fire” [4]. As a result, many more buildings burned than necessary, which only served to accelerate the rate of the fire and exacerbate the damage caused, and resulted in many more homeless victims.

One thing to keep in mind about the fire is that due to the fact it was during the aftermath of an earthquake, many key functions of the town were already in disrepair. One of the most devastating problems caused by the earthquake was the rupturing of the city’s water mains, not

only depriving survivors of waters, but severely crippled any efforts to fight the fire itself. As a result, many buildings that did survive the earthquake relatively intact, such as the Palace Hotel, were lost to the fire instead. Many other areas of significance were destroyed as well, ranging from laboratories containing hundreds of botanical specimens to the original Californian flag. While all of this may seem bad enough, simply hearing about the damage is much different than fully understanding the numbers of how much was really lost.

So how bad was the earthquake of 1906 truly? The earthquake itself affected around 375,000 square miles, although half of which were in the Pacific Ocean. It ruptured the surface for about 290 miles along the San Andreas Fault. For comparison, the earthquake that happened in 1899 covered an area of only twenty-five miles or so. The rupture's speed was estimated to be nearly six thousand miles an hour, and it's estimated that the ground shifted between four and five feet a second. The property damage was estimated to be over four hundred million dollars, which in today's age would be more akin to eight billion dollars of damage. This earthquake was also significant because it was the world's first natural disaster to have its effects recorded by photography [5].

The earthquake also had massive repercussions on the city as well. It is estimated that out of the city's population of four hundred thousand, two hundred and twenty five thousand were homeless. Over half of a city's population lost their homes within a span of three days. The fires that caused most of the destruction destroyed around twenty-eight thousand buildings in an area of five hundred city blocks. This was approximately one fourth of the city. In the first nineteen months San Francisco spent ninety million dollars on recovery. The army was called in and helped set up twenty one refugee camps for the survivors [6]. The estimated amount of people that lived in these camps was that of about twenty thousand.

The first quoted figures about the amount of casualties were around seven hundred people. However, this was a severe underestimation and did not take into account immigrants, so the real figure is much closer to something over three thousand people. Many of the survivors were living in relief for as long as up to two years after the events of the earthquake as well[7].

The earthquake of 1906 had two types of effects, immediate and long term. Immediately, aside from the loss of life and home, the city was thrown into chaos. Looters ran wild, people were setting fire to their own homes to try and salvage some value out of them, and people were dying in the streets. The army was called in and was a key figure in relief efforts, patrolling the streets to attempt to discourage looters, guarding important buildings that still stood, like the mint, jail, and post office, and feeding and sheltering the multitudes of people who had been affected by the earthquake. However, on April 18th, due to the high amount of looting that had been occurring, the mayor declared that “The Federal Troops, the members of the Regular Police Force and all Special Police Officers have been authorized by me to kill any and all persons found engaged in Looting or in the Commission of Any Other Crime” [8]. It has been estimated that as a result of this declaration, over 500 people were shot for looting, although there are suggestions that many of them were not looting at all, and were simply trying to recover things from their homes. The army also fell under criticism due to the fact that many of them had been accused of looting as well.

In terms of long term effects, the earthquake has played a significant role in earthquake prevention codes. Immediately following the earthquake, codes became so strict and thorough that many places did not have as advanced protections all the way until 1950. Plans to rebuild the city were immediately formulated and enacted, and in 1915 an exposition was held to celebrate the city’s recovery.

Since 1915, the disaster has been commemorated annually in San Francisco. Survivors and loved ones gather at Lotta's Fountain, a landmark that served a key role in the trading of information during the earthquake. And we are attempting to use our game as another way to spread knowledge and awareness about the 1906 earthquake. This is not the only time a game has been related to a historical event, however.

Games, like all forms of media, can be referential. Being referential, they can refer to events that have actually occurred, for either education or entertainment purposes. Games have also referenced historical events either to inflame such memories back into the public eye, such as the example of Super Columbine Massacre RPG or September 12th. We have used these games as guidelines of games we would like to be similar too, and games whose example we strived to follow.

Methodology

Organizational Approach

When we started to design 1906, our initial plan for how we would go about designing the game was much different compared to our final approach. Initially, we had planned to create all the assets needed, albeit in rough form, in A term, while technically we would attempt to have a finished framework as well. Our schedule in B term was then to proceed to create finished versions of all our art assets, and have a first playable created as well. In addition, we hoped to start contacting voice actors so we could begin recording voice in early C term. C term was then

simply scheduled for polishing up the art and the gameplay, and increasing the production value as much as possible. While this was the initial approach we were to take to create 1906, it is not the design method we followed.

Art Design

In terms of art design, we had always envisioned the game as realistic as possible. Due to the gravity of the subject matter, we did not feel that it would be appropriate to try and portray the game in a humorous manner, and we also did not want to make the game too symbolic either. And so we strove to be as true to real life as possible. And the most important way of accomplishing that goal was using references.

References were used everywhere they could be. The initial concept art conceived for this game was based off of reference photos of the event, and the clothing of the characters were based off of realistic period wear. All of the buildings were based off of real buildings, although some modifications were made. Initially we had planned on basing each level off of a specific part of the city, but we ended up finding that too restrictive and tried to make something that felt right, as opposed to being completely accurate, as to not stifle both gameplay and design. A game that we derived some visual reference from was *Red Dead Redemption*[9]. We felt that this game had not only an art style that we could possibly reproduce, but it also very closely resembled ours because it is set in America in 1911.

In addition to references, another art design concept we planned on using was to create assets that could be easily palette swapped. When we were initially designing the man and the woman models, we intentionally left off shoes and head gear, with the plan being that we would have different shoes or headgear in order to make our characters look different without using completely different models. A process was created that allowed us to place objects in certain locations based on the skeleton of the model early in A term, and while we did not take advantage of this feature as much as we had initially expected to, it is still the method used to place items on the models.

Our initial design for houses was to be a very similar approach as well. The first possible idea for house development, which never ended up coming to fruition, was to create several possible blocks of a house, allowing us to procedurally generate a house by picking from several possible roofs, second floors, first floors, or other pieces. The second idea which was followed a little more closely was to create houses that could have different textures applied to them. This way even though the shape of the house would be the same, it would still look noticeably different, possibly more burnt or in better condition than other houses in the neighborhood. However, even this did not end up happening to the extent we hoped it would.

While we had hoped to have a variety of different looking houses and people, we also realized that this approach was not as strictly necessary as we thought it would be either. Once we began building our levels, we realized that while there may not be as many different houses or people as we had wanted, by use of lighting and placement we could still achieve the effect of appearing to have an actual livable world.

In order to make our world feel more believable, we decided to also create believable reasons why the player could not leave the boundaries of the level. As opposed to invisible

walls, we used rubble, debris, and other objects to block the player in. Buildings were placed in such a way that while there would be neighborhoods that looked like they could be lived in, there would be no easy way for the player to leave either. Fire was also used as a deterrent to the player to prevent them from leaving. In addition, to the fact that we did not create a world that extended in all directions, we used many tricks such as the placement of buildings and the creation of walls and fences to make it so that the player could at no point peer out into the wild blue yonder.

Tech Design

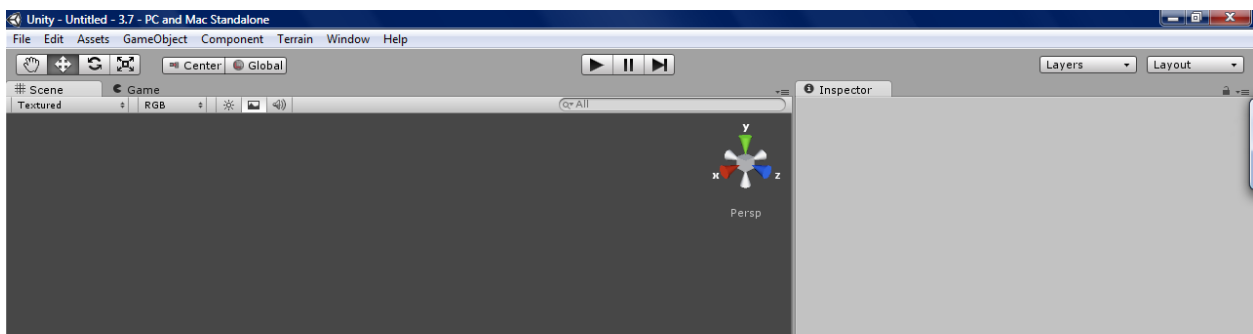
The technical approach to 1906 was steeped in complexity. To combat this, several tools and design strategies were picked to combat any muddling details. Unity was used as a platform for development, C# was picked as a language and Singletons were used to handle most functionality. Code was edited and tested from several editors with varying success, but UniSciTE was finally chosen as the sole editor.

After design strategies had annealed and the accompanying tools had been picked, the code structure of each aspect of the game was hammered out. Our first singleton was made to handle the three main variables representing the status of the game (Health, Hope, and Morale), and was named StatusSingleton. A singleton was assigned to handle the logic for the Heads Up Display (HUD) and Graphical User Interface (GUI), known as GUIcontroller, as well as Singletons for specific HUD components like the Nav Arrow. A singleton was also made for handling any interacting logic to all NPC citizens, known as PeopleHandler. Finally, a Singleton

named EventHandler was made for handling the overarching logic of the game such as quest events, pausing, and scene change functionality.

Unity Platform

The Unity Game Engine is a 3D cross-platform game development tool created by Unity Technologies [10]. It is capable of publishing to PCs, Macs, Current-gen Consoles, Mobile devices, and a web player without almost any reconfiguring from the developer side. Unity was chosen for its support for scripting languages like C#, its resultant affinity for prototyping, and its reputation for usability for both people with a strong technical background and those without. Perhaps the most advantageous aspect of Unity, however, was its content management system. In other game engines, importing assets is a long and arduous process, often involving monolithic content pipelines and hand-coded exceptions. With Unity artists can simply drag and drop their assets into the project, where it will automatically be converted to proper format, organized, and stored. Due to the volume of required assets for this project as well as other demanding technical tasks, having this automated system was a boon.



C#

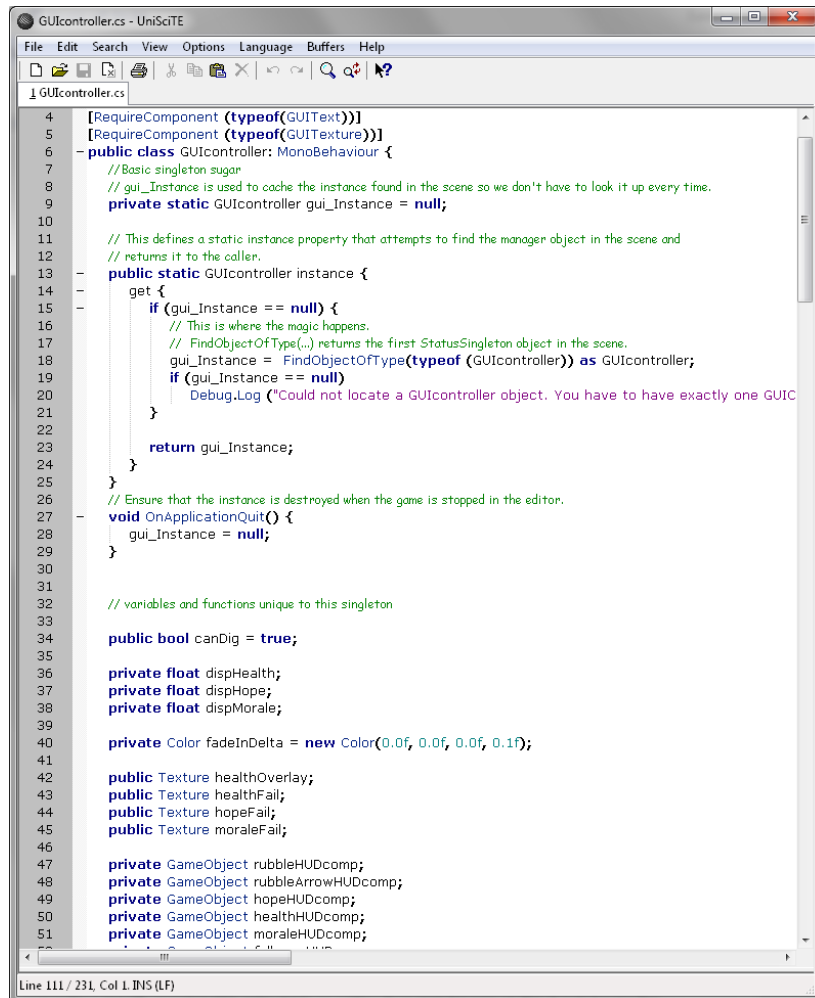
The C# Scripting Language is very versatile. Like all scripting languages, it is very easy to compile and omits a lot of syntactic sugar that exists in lower level languages like C++ or Java. This lack of syntax, while advantageous for rapid prototyping, also results in the code being vague in certain areas. Performance also takes a hit in regards to framerate and maximum simultaneous components, as is true with all high level scripting languages, but optimizations were enacted whenever possible to resolve this easily foreseeable issue. Such optimizations included the use of precompiled libraries and functions that were written in lower-level languages like C and C++. These libraries were built into the Unity framework and were completely uneditable, though were able to be called by higher scripting languages and files that we could edit. Without these precompiled functions it would have been impossible to accomplish what we did in Unity.

Singletons

Much of the functionality within 1906 was based on singletons governing each of the main aspects of the game. The general advantage of singletons is that for aspects of a game that only exist once, like a HUD or a player's game statistics, treating those aspects as singletons will make them accessible and editable from anywhere else in the code (including from other singletons). Many code design strategies eschew singletons, as they are widely regarded as an inelegant solution to programming problems, due to their propinquity to lead to circular dependencies in code. When handled properly and sparingly, however, singletons are a plausible and functional solution to programming structures.

Furthermore, the distributed nature of the Unity game engine creates an environment with an affinity for singletons. Within the Unity environment there is no unified codebase, as much of the unity code is proprietary and unviewable. Rather than adapt to this "organization" technique, the use of singletons allowed our code to be more independent and insertable, which was a boon in dealing with the complications between Unity and the computer systems that followed during development. A breakdown of those complications can be found in the Development section.

UniSciTE



```
4 [RequireComponent (typeof(GUIText))]
5 [RequireComponent (typeof(GUITexture))]
6 - public class GUIcontroller: MonoBehaviour {
7     //Basic singleton sugar
8     // gui_Instance is used to cache the instance found in the scene so we don't have to look it up every time.
9     private static GUIcontroller gui_Instance = null;
10
11
12     // This defines a static instance property that attempts to find the manager object in the scene and
13     // returns it to the caller.
14     - public static GUIcontroller instance {
15         get {
16             if (gui_Instance == null) {
17                 // This is where the magic happens.
18                 // FindObjectOfType(returns the first StatusSingleton object in the scene.
19                 gui_Instance = FindObjectOfType(typeof (GUIcontroller)) as GUIcontroller;
20                 if (gui_Instance == null)
21                     Debug.Log ("Could not locate a GUIcontroller object. You have to have exactly one GUIC
22             }
23         }
24     }
25 }
26 // Ensure that the instance is destroyed when the game is stopped in the editor.
27 - void OnApplicationQuit() {
28     | gui_Instance = null;
29 }
30
31
32 // variables and functions unique to this singleton
33
34 public bool canDig = true;
35
36 private float dispHealth;
37 private float dispHope;
38 private float dispMorale;
39
40 private Color fadeInDelta = new Color(0.0f, 0.0f, 0.0f, 0.1f);
41
42 public Texture healthOverlay;
43 public Texture healthFail;
44 public Texture hopeFail;
45 public Texture moraleFail;
46
47 private GameObject rubbleHUDcomp;
48 private GameObject rubbleArrowHUDcomp;
49 private GameObject hopeHUDcomp;
50 private GameObject healthHUDcomp;
51 private GameObject moraleHUDcomp;
```

Line 111 / 231, Col 1, INS (LF)

Figure 2: UniSciTe Screenshot

UniSciTE is Unity's proprietary code editor that ships with Unity. It has the capability to syntactically recognize C#, Boo, and Javascript; the three scripting languages supported in Unity. It also has features such as predictive auto-complete for variable and function names and syntax coloring for system functions and variable types. It is lightweight, so that many files can be opened and edited at once, and has a tab feature, so it does not take up much screen space. There were different advantages offered by other code editing applications, their comparisons to UniSciTE are in the Development section.

Singleton Dependency Outline

StatusSingleton stands at the top of the hierarchy, drawing information from nothing, but alterable from anywhere. All of its functions and variables revolve around the three main status variables of Health, Hope, and Morale. Hidden variables were declared keep each status variable in a prespecified range, as well as publicly accessible functions to alter the data in specific ways. For purposes of security and in accordance with proper programming practices, the master variables of Health, Hope, and Morale are never altered directly. This prevents any function anywhere from accidentally assigning an improper value to these variables and breaking the game as a result.

GUIcontroller takes the values from StatusSingleton to alter the components of the HUD. Each of the three main variables has a different GUI component associated with it. Morale, for example, is represented by a burning city. As Morale depletes the city burns more and more until the entire city is consumed. Hope is represented as a silhouette of a man, whose posture and movement display the amount of hope that is left in the player. Health is represented as a red overlay, changing on alpha level and becoming more visible as the health of the player becomes

lower. Other GUI components include a dynamic lose screen, which display an overlay over the entire screen, as well as 3D components, which are described in the art development section.

PeopleHandler also draws from the StatusSingleton, namely reading in the health variable and performing logic from it. To summarize, when the health variable reaches zero, the main player object position is changed to that of a different NPC, and a corpse is spawned at the player's previous location. Additional logic concerning pathfinding for each walking NPC on is also handled in PeopleHandler, though the navmesh itself is still generated and handled by the imported A* algorithm from AngryAnt. A description of AngryAnt's A* Pathfinding can be found in the development section [11].

The EventHandler Singleton takes from the Status Singleton also, as well as the PeopleHandler, to manage quests and overarching level logic. In EventManager the pause functionality is handled, which is setting a global variable in Unity to not call fixed update functions of any other scripts. This prohibits certain functionality that could otherwise be possible, but such functionality was not required for the context of this game. Also in event manager every quest script is stored and called. This allows certain quests to be made available after a certain amount of time has elapsed, and also allow quests to share information like NPCs and items.

Sanity checks and other verifications were placed throughout the singletons to ensure that all variables were in their proper ranges and no conflicting or otherwise bad data was being passed to any functions anywhere.

Organizational Design

In addition to both art and tech, we also started trying to design how the game would actually be played very early on as well. As stated earlier, the initial design of the game was that of a game that would start as a first person game, then switch to a third person game, then switch to a real time strategy style game. When we quickly realized that the scope of a project like that would be absolutely enormous and much too difficult to undertake in such a short period of time, we decided to attempt to dial the scope down.

The next plan for our game was that of a game that would put morality in the hands of the player. During the earthquake and fire of 1906, many of the citizens in the city ended up committing morally questionable acts, such as setting their own houses on fire due to having fire insurance but not earthquake insurance, or even plain rioting and looting. The state of the city got so bad that the mayor called in the National Guard and declared a state of martial law, including the riot for anyone to be shot upon being seen looting, even if it was their own property they were trying to recover from. And in our second idea of how the game would play, we wanted to let the player experience all of these possible events.



And so in our second concept of the game, the player would be placed in a city in the midst of destruction, and left to choose what they wanted to do themselves. It would be up to the player to decide if they wanted to help out their neighbor, so to speak, or simply gather their own supplies and walk away. Or as an alternative, the player could think only of themselves, and do such things as set their own house on fire, or even loot from their neighbor. Due to the way we had planned on implementing gameplay, it would be very easy to make multiple different actions reward the player in one specific way, despite the possibility that one of the actions was helping your neighbor and another was setting your house on fire.

However, we again were very quickly able to tell when our scope had grown to large, and allocated many of those possible features to features that would be nice if we had the extra time, but not necessary to achieve functionality. Instead we decided to focus on having three separate days, meaning three separate levels, for the player to try and survive and help out as much as possible. We were not able to create levels for two of the days, but the initial concept of trying to help people remained in. We also were able to get in all the key mechanics we had planned on in the beginning.

Development

Our team's composition led to us having to approach making 1906 in an atypical way. We had three artists to work on the art, and so ended up spreading the creation of art greatly around, with focuses on character design, asset creation, and animation. But there was only one full fledged programmer on the team, which meant that while we could make many art assets simply for polish and to make the game look better, the essential features of tech simply had to be done.

Unity 2.8 to Unity 3

The transfer to Unity 2.8 to Unity 3 was a time consuming but necessary transition. On one hand we had the disadvantage of refactoring the code, but on the other we had faster compile times and better additional features to apply to our game. One of these advantages was Ambient Occlusion Culling.

Ambient Occlusion Culling

Ambient Occlusion culling was one of the advantages we got from switching to Unity 3. Its function was primarily optimization, allowing more geometry to be displayed at one time on

the screen, giving us the ability to display a more vivid world without the game slowing down. The drawback to this built-in feature of Unity 3 was the time it took to compile the occlusion data, which on top machines still took six to eight hours, and crashed regularly. Also, since it was a proprietary feature, there was no way to optimize the procedure by using graphics card support, or to even see what was going on in their code. After getting it to work, however, ambient occlusion became a vital optimization to our game.

MonoDevelop

MonoDevelop was an alternate script editing application that was considered in place of UniSciTE. It could be tied to the Unity project quite easily, and had a much more navigable interface along with better automated code completion. Repeated and unexplained crashes and lost progress caused the tech to conclude that it was more trouble than it was worth and resumed using the built in code editor UniSciTE.

Third-Party Scripts and Libraries

AngryAnt's A* Pathfinding

Emil Johansen, otherwise known as AngryAnt, is an employee for Unity Technologies who specializes in AI development in his spare time. He offers on his site free of charge several

tutorials and unity projects of fundamental AI practices. One of the libraries that he offers is a navmesh generation tool that attaches itself to unity, as well as a rudimentary entity that can navigate the generated navmesh. AngryAnt's library was selected for its ease of insertability and compatibility with the unity system, as well as its free-to use license. AngryAnt has recently updated his website and his licensing agreement, but at the time we acquired the unity project the license was freely available.

Flickering Fire Script

To help make our fire look more realistic in game, we needed to handle the way it gave out light realistically as well. An openly licensed script for handling flickering lights created by fires was found on http://www.unifycommunity.com/wiki/index.php?title=Flickering_Light, and we proceeded to use this in all of our fires in the city, both inside and outside.

Unity's Procedural Animation Package

Unity's Procedural Animation package was heavily considered as a feature, as using it would cause a much greater sensation of immersion in the player. Upon implementing it, however, the amount of additional processing required for a crowd of people was too much to the system, and so the feature was dropped.

Art Development

Our approach to developing art assets ended up being much different than the original idea we had envisioned. Initially, we had planned on attempting to create all of the rough assets in the game in A term, all the finished assets in B term, and polishing everything even more in C term. But even in the beginning of A term, we realized, thanks in part from some advice from our advisers, that it would be more beneficial to focus on creating certain polished assets first, specifically the ones that would be seen the most often. This would allow us to create professional looking assets, and give us a large berth of time to make sure that they actually worked in game. Following this plan, the majority of A term was spent fully designing two character models to be in the game, complete with texture, normal map, and bone structure, and several houses to be included as well. Several rough props and an initial walk cycle were created as well, and were used extensively in our initial demos of the game.

The beginning of B term was then spent adding additional polish to all of our previous assets before moving onto other things. Both the man and woman were fully put in game, and newer animations were created as well. Fire was created, so we could start using it in the game world, although it wasn't fully finished until C term. We also worked together with the tech side to create the assets that were needed for quests. Our house designs were finalized and implemented, and by the end of the term we had a rough, explorable neighborhood with all of our assets.

C term was used for mostly level design and world building, with an additional focus on polish. The level for our game was built out of the assets we already had, and we created working floor and sidewalk textures as well. More time was spent on the implementation of the assets than the creation of new ones, however. Lighting was another element we spent much time working on, as while we had had things like lamp in game for a while, we did not have it looking like it was actually emitting light. Similarly we had houses that were on fire, but not in a way where the fire looked like it was actually burning inside the house. Finally, we simply had to fully populate the city, blocking off areas that the player was not supposed to explore.

General Asset Workflow

Though each artist had their own personal workflow that was tailored to their needs, the general workflow for most of the standard assets followed the same path. The models were started in a 3D modeling software such as Maya or Blender [12]. From that starting point the artist would block out the basic shape of the asset using extrudes and the basic manipulation tools provided by the software. The goal at that stage was to create a functional and believable mesh that resembled the real life item, while at the same time keeping the mesh at a relatively low polygon count.

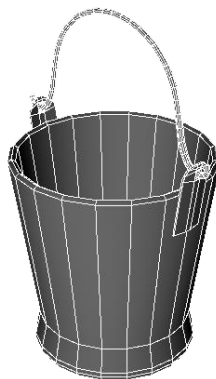


Figure 4: Wireframe of the bucket item

Once that was complete the next step was to create a well organized and planned out UV map for the asset. One way to ensure that the UV map would lend itself to a believable texture was to apply a checkered pattern to the asset before creating the UV map. By looking at the checker pattern on the asset the artist was able to discern if the UV map was properly scaled on all of the different parts of the mesh. This ensured that each area of the mesh received the same amount of texture detail.

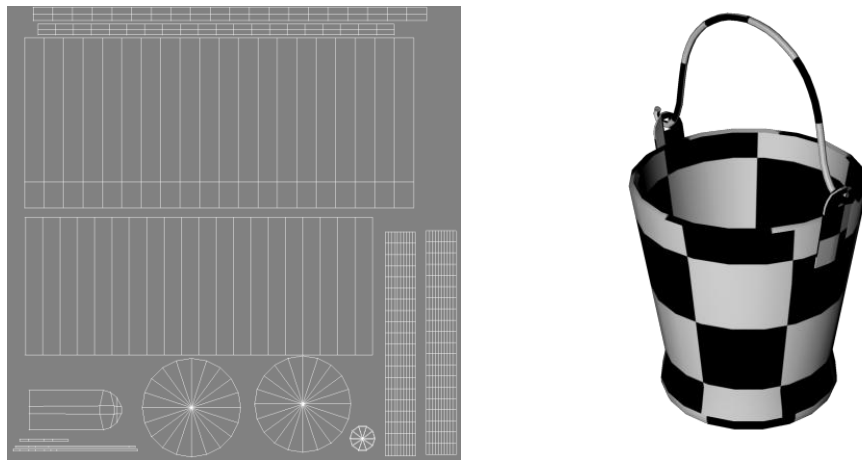


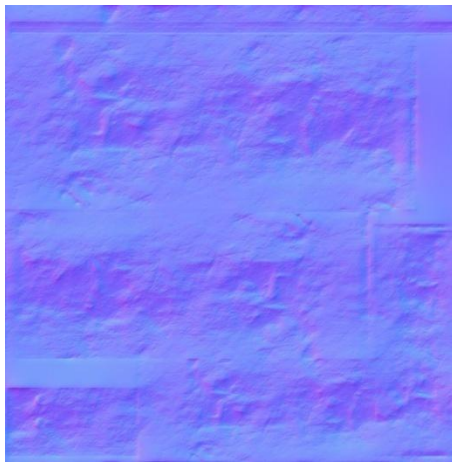
Figure 5: UV Map and Checker Pattern

After that the next step was to bring the UV map into Photoshop to begin creating a diffuse map [13]. That was accomplished by using Maya's "Create PSD Network" command, which brought the UV map from Maya into Photoshop at the proper size in an easily editable way. By saving the Photoshop file after changes the artist only had to hit the "Update PSD Network" button in Maya and the texture would automatically be imported onto the mesh. Many of the textures that were used were obtained from the site CGTextures.com [14].



After the diffuse map was [Figure 6: Diffuse Map](#) maps could be made from it.

The program CrazyBump was a great tool for creating amazing normal maps quickly and precisely. The program would load the diffuse map and provide two options for a normal map. The two options were the inverse of each other, and the artist would have to choose which one properly displayed the desired bump direction. After that the artist could adjust the image to fine tune the details.



[Figure 7: Normal Map](#)

The next step was to create a specular map to give the bucket a nice shine. There were two ways to create a specular map, each for the different ways Unity and Maya handled them. Maya required a black and white image to create the specularity, and the program CrazyBump was also used for this. Unity on the other hand requires that the specularity of an object be held in its

diffuse map's alpha channel. In order to create the specularity in a way that Unity could handle the artist would select a region of the image that would make sense to be shiny. In the case of the bucket example the Select Color Range tool was used to select all of the whiter areas of the image, and then with them selected an alpha channel was created to hold these areas.

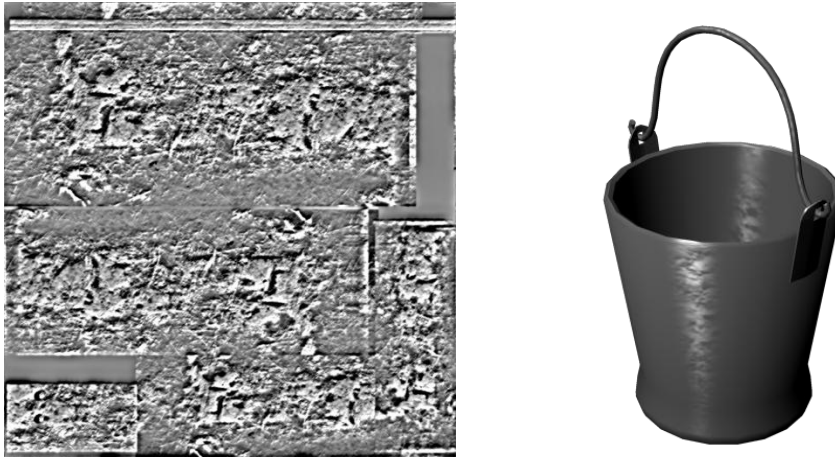


Figure 8: Specular Map

With all of these different maps applied at the same time, a more accurate representation of the real world item was attained.



Figure 9: Final Render

Once the diffuse, normal, and specular maps were made the final step was to bring the assets into Unity. One important but simple step not to forget was to delete the history and freeze the transforms of the model. Often times if a model wasn't displaying correctly the simple answer was that deleting history hadn't been done. The great advantage to using Unity as the game engine was that importing art assets was as easy as dragging and dropping the folder into the program. Once imported the artist only needed to drag and drop the textures into their correct slots to complete the process.

Character Workflow

The workflow for creating a character was much different from general asset creation. The characters were sculpted from high polygon models and then brought down to a level that the game engine was able to handle. The characters were based on a base mesh that came with the program ZBrush [15]. From this starting point unique characters were slowly built up by using ZBrush's different built in brushes, alphas, and strokes.

Once the artist was happy with the way the high poly model looked it was time to create a UV set. ZBrush has an excellent plug-in just for this purpose, and it is called UV Master. Once open the artist would create a clone to create the UV's on. To ensure that seams didn't appear in less than ideal places, the artist would paint onto the mesh areas where there shouldn't be a seam and areas where it would be ok to have a seam. In general the face was painted to be kept together and a line was painted down the back to allow for a seam. Once this was complete the UV's could be unfolded and flattened for the artist to see. Once happy with the results the UV's

can be copied and pasted back onto the original mesh. After that the artist then polypainted the model to create believable skin tones and clothing.

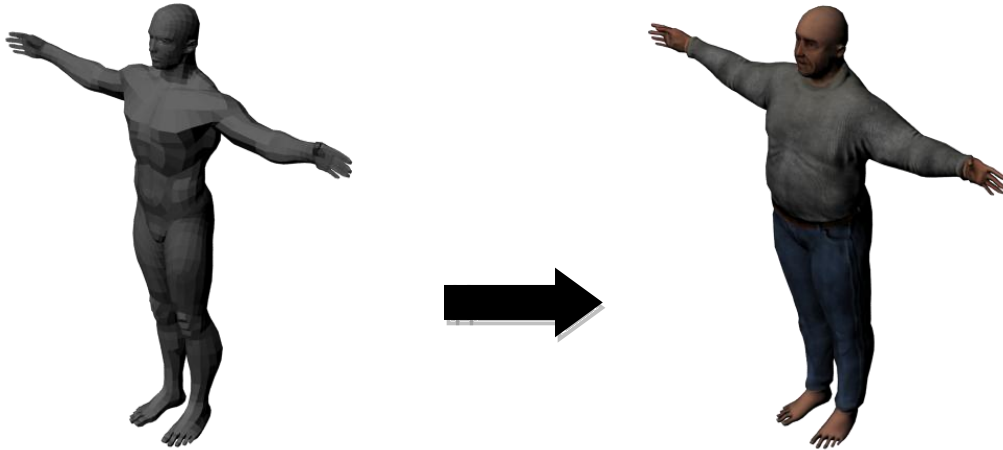


Figure 10: Base Model to Final Model

Once this is complete the model is ready to be brought into Maya. By using the GoZ feature in ZBrush the model will automatically be opened up in Maya ready to edit. In order to export the diffuse map and the normal map the Multi Map Exporter plug-in was used to ease the process. By simply highlighting which maps were to be exported ZBrush would create properly oriented and sized texture maps. Another bonus to this plug-in was the ease in which it produced ambient occlusion maps. When ambient occlusion was made in Maya using shaders there would often be a dark line along the UV map seam on the model. When the map was exported using the plug-in no dark line appeared on the model.



Figure 11: Diffuse, Ambient Occlusion, and Normal Map for Character

The next step was to create a rig onto which all of the characters would be mounted for animation. By keeping all of the proportions similar on the models the same rig was able to be used for most of the time. Some complications were encountered with the woman's dress where extra bones had to be added to make sure the legs did not go through the dress. The rig was made first by placing a root joint at the hip and expanding to each leg, up the spine, and then to each hand. The mesh was then bound to the skeleton using a smooth bind. The next step was to paint weights so that the mesh would react more naturally to being bent and rotated. IK handles were actually not used and the animations were entirely made from joint rotations. The animations for each character consisted mainly of a walk cycle as well as an idle animation.



Figure 12: Rigged Model

The final few steps were to bring the model and animations into Unity and split up. Getting the model in was as simple as dragging and dropping into Unity, and then from there the correct diffuse and normal maps were assigned. The final step was to split the animations from the single animations based on their key frames.

Workflow Complications

During the creation of the assets there were a few areas in the workflow that needed to be modified to take into account issues that were occurring with some of the assets. One common problem that was showing up was reversed normals on some faces of the objects. In Maya faces with reversed normals show up correctly, but once brought into Unity there was a definite visual anomaly where an object would appear to be inside out. Often reversed normals occurred on objects that had been made using Maya's extrude tool. The solution to this problem was very simple, with the problem being solved by simply selecting the offending face and clicking the reverse normals button

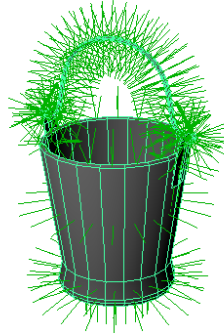


Figure 13: Normals for Bucket Item

In terms of character workflow issues, the female character proved to have quite a few issues. When the character was originally being developed the intent was to have swappable heads to be able to create a wide array of characters by simply changing out the heads. For this reason the woman model was created without a head. The woman model was also created from the same low polygon base mesh as the man, but the woman also needed a dress. This required a lot of manipulation of the polygons and thus changed the topology of the model greatly. In order to fix this the retopologization process was done on the woman. The process worked for almost the entire model, but the hands did not come out correctly. This needed an alternate solution, and a creative work around was found. Since the main reason retopologization was needed was because of the dress and not the hands, the original low polygon mesh with the dress attached was used. It was imported into ZBrush and then the high polygon mesh details were projected onto the low polygon model with better topology.

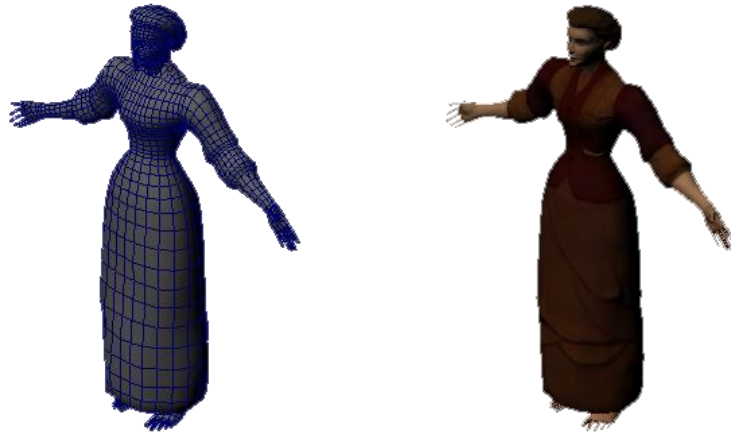


Figure 14: Female Model

Another aspect of the woman's dress that caused issues was during rigging and animating. The original rig created for the man did not take into account the extra polygons that the dress needed, so the dress was unaffected by the rig and the legs went through the dress. This required going back into the rig and applying two more bones to the root joint, each of which handled the dress and kept it from going through the legs.



Figure 15: Modified Female Rig for Dress

Flame Development

Another one of the key artistic features we had in game were the fire effects covering the city. We knew from an early stage that due to how the situation played out in reality, our city would need to be covered by fire. Since this would be such a key aspect of the game, special detail had to be taken to make sure that this ever present feature looked as good as the rest of the city.

The process for creating realistic looking fire was deceptively simple, however. The first step was the creation of the smoke and fire textures themselves. A single frame smoke texture was created in Photoshop using the default filters available, and an eight frame animated fire texture was created using mostly the same method. Once both the textures were created, they were ported into unity, and particle systems to display the fire itself were created.

Every fire in the city is comprised of three different elements. The first is the particle system for the smoke, which is set to travel upwards into the sky, gradually disappearing over time. The second is the particle system for the fire. This is slightly more complicated than the smoke particle system, as not only is the fire texture animated, but a much more sporadic movement pattern was needed for the fire itself. In addition to that, while the smoke texture only needs to be told that it has an alpha layer in unity, the fire texture needs to be blended using the soft additive particle effect, and has to be colored as well in the engine to help provide a more realistic flame. The third element in every fire is the actual light the fire is emitting, which is an orange colored pointlight with the flickering light script attached to it. Every fire in the city is

made up of these elements, and while several of the variables are changed for specific fires to create noticeable differences, all of the fires are cut from the same original cloth.

Heads Up Display

When creating the heads up display (HUD) a lot of inspiration was taken from the game *BioShock*[16]. After looking at screenshots and playing the game it was decided to implement a similar system. Two icons were to be placed in the lower and upper left corners and the player characters arm would fill the lower right. To keep the HUD readable they were created to be very iconic and simple. For this reason they were kept black with a small amount of white to give some contrast to the actual game. The hope meter was placed in the lower left and was made to look like the San Francisco skyline with the words “Welcome to San Francisco” written in front of it.



Figure 16: Hope Meter

The level of the flames behind the city indicated on how well or poorly the player was doing in terms of hope for the city. The larger the flames the lower the hope of the city and the worse the player is doing. The upper left corner houses the moral meter which was represented by a silhouette of a man. That asset was created by applying a toonshader to the second male

character with a black fill and white outline. When the player is doing well the character is standing giving a thumbs up, when the player is doing only alright the character slouches forward and puts his head down, and when the player is doing poorly the character kneels down and cries.

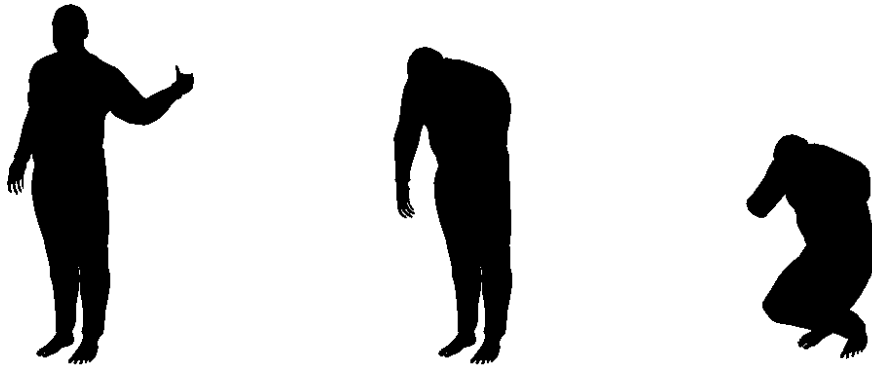


Figure 17: Different Stages of Moral Meter

The final two HUD elements were the navigation arrow and the mini game slider. The navigation arrow was used to guide the players toward their next objective if they became lost and appeared in the top center part of the screen. The mini game slider was used as a gameplay element where the player needed to press a corresponding button at a certain time to make something happen. An example of that was having to hit a button at the correct time to make the player dig through rubble to find items. Both of these HUD items were created in a similar style. Inspiration was taken from an old fashioned wrought iron look to try and fit in with the setting of 1906 San Francisco.



Organizational Development

Organization was something that our team focused on from the beginning. While the plan for the game changed multiple times, we made sure that we always knew exactly how we were going to proceed in creating the game. From the beginning of A term, we started by creating asset lists for both art and tech, trying to focus in on what we thought was most important. That is not to say that every method we used worked, however.

In B term we tried using a website called Pivotal Tracker to keep us organized. The purpose of pivotal tracker is to allow users to put all the tasks that are required to be done for a project online, assign them a value of how long the task should take to complete, place it in a category based on what type of task it is, and let users assign themselves these tasks. All of this information is displayed to all other users, so someone can see exactly what everyone else on their team is doing, in the hopes of preventing superfluous work. In addition, the website tracks how long it takes for each task to be completed, and uses that value to help create an estimation of how many hours of work will be done in the week, and allowing users to see how many tasks that the program estimates they can complete in a week. But unfortunately, our team was not able to truly use Pivotal Tracker to its fullest.

In reality, Pivotal Tracker was simply not designed for a team like ours. Pivotal Tracker functions best in an environment where people are working from a code repository, and so are constantly checking the website as to make sure they are working on something someone hasn't already done. For programmers, pivotal tracker is great because it allows you to divide up what code needs to be written for what parts of the project, and so even if someone focuses on a

certain aspect of the project, they can still see that something in another area needs to be done and assign themselves to do it. This didn't work on the art side for two primary reasons. The first is that while we could divide the art up into sections, there was no real advantage to doing so. While we could say we had both a task in animation and a task in asset creation to do, it didn't really matter who did what or in what order even, as those are two very separate and almost unconnected jobs. The second reason is that we simply did not check it enough. While it was always available to look at, people didn't look at it reliable enough to use it as the primary source of what people were doing, so our team still ended up sending emails out saying what part of the game they were working on, to prevent people from doing the same work. And on the tech side it might have been slightly more useful for a team, we had only one member writing code, and so while he could use it to organize his thoughts, there was never any need to worry about people writing redundant code. Because of this, while we tried using it during B term, we ended up phasing out Pivotal Tracker as something that wasn't a bad experience, just not one that made our lives any easier.

What our team did use to keep ourselves organized throughout the entire project was two major methods. The first was a constant stream of communication. A mailing list was created before the school year even began, and emails were sent daily, along with weekly meetings both with our advisors and among the team. People would always communicate what they were working on and what they thought was necessary to be done, and offer suggestions or help on any needed subject. The second method we used was a constantly updated repository of all our assets and code, located on the website Dropbox. Everything that anyone created, no matter how simple the asset or how few lines of code was placed on the website, and as a result all team members always had access to the latest resources. Dropbox became especially useful later in

the project, as it allowed us to always have the latest build up online, allowing anyone to access it from any computer.

Conclusion

Our finished game is not at all what we expected it to be. We started the year with ideas of developing some complex real-time strategy game set in a realistic depiction of San Francisco, complete with moral choices and multiple endings.

The biggest lesson that our team felt we learned from this experience is an understanding of scope. Throughout the entire project, we constantly had to reevaluate what we thought we could do with what we wanted to do. There was simply not enough time for everything to be done, and our project became a very complex balancing act trying to figure out what we could cut and what we wanted to keep.

That is not to say we are not proud of what we accomplished. We feel that what we managed to achieve was momentous, in terms of both art and tech. However, there is always the desire for more. We understand now what things we spent too much time on and what things we did not focus on enough, and if we were to attempt this project over again, we feel that we would be able to create much more in the same period of time thanks to what we learned.

Appendices

Appendix A Art Assets



Figure 19: First Person Arms, Bandage, Bench, Boot

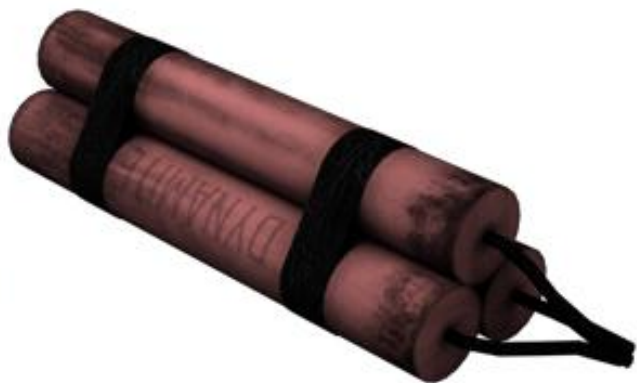


Figure 20: Bucket, City HUD, Compass, Dynamite



Figure 21: Cart, Girder, House 1, House 2



Figure 22: Broken House, Man 2, Photo Album, Lamp Post



Figure 23: Pile 1, Pile 2, Pile 3, Plate



Figure 24: Table, Cup, Trolley, Woman

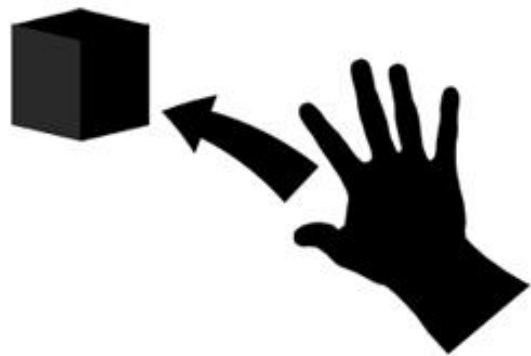


Figure 25: Cursor Icons; Take, Give, Dig, Put Out Fire

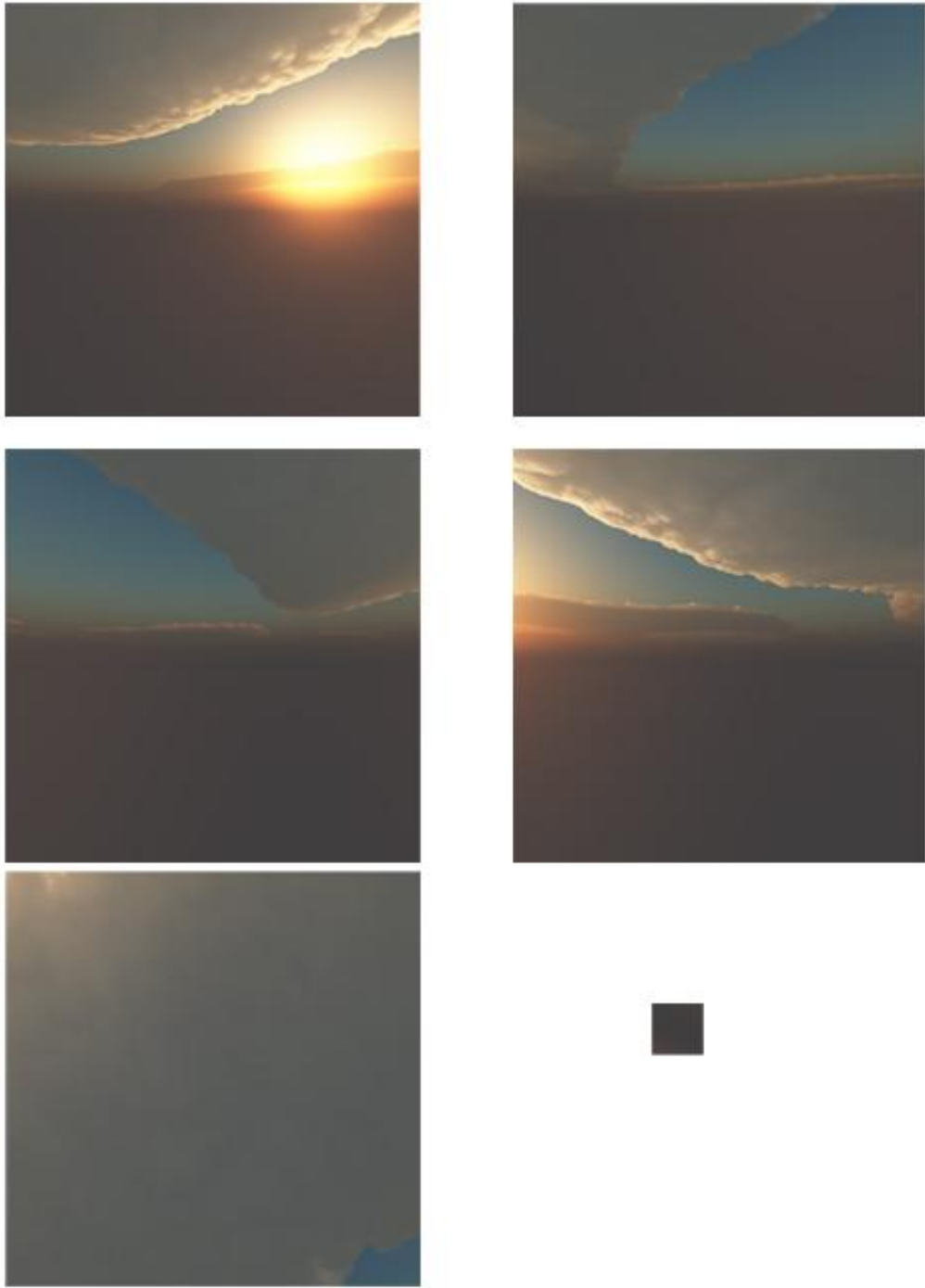


Figure 26: Skybox Images

Appendix B Script Index

1906 Script Index

DAY 1 QUESTS

1. Our House (In The Middle of the Street)
 - a. Right as Rain
 - b. Save the Memories
 - c. Where To?My Father's Watch
Approaching Flames
The Arsonist's Gambit
Righting Ambulance
- . Find a Fulcrum

•

DAY 2 QUESTS

1. My Kingdom for a Teacup
 - a. Find three teacupsPump It
- . Find a pump
 - a. Find a BucketSoup Line
- . Get Water for Dough
 - a. Scrap Wood for Fire
 - b. Bring Bread to FamilyRighting Streetcar
- . Find a Fulcrum
- Army Now *Talk to Britt about it
- . Place proclamations around

DAY 3 QUESTS

1. Dynamite the buildings
 - a. Dynamite
 - b. Explosion FrameworkFire Fight
- . Bucket
 - a. Water splash particlesLeavin' on a Steam Train

Four meals away from Anarchy

- . Bread
- a. (Horse?) Meat

Barks:

<https://docs.google.com/document/d/1aTC88Eh1MXsz1YDYXriXGHT3IL62KeRd04eNNDd uHHM/edit?hl=en&authkey=CN2ZwJkJ>

Vignette:

<https://docs.google.com/document/d/1ByJ3OwQ6afKcZvRM2tdw1AiNgYSXSA8VXIbBetE7C3s/edit?hl=en&authkey=CJWsuNkN>

Intro Cutscene to Day 1

Narrator:

1906, April 18th, 5 AM. The world, America, California, San Francisco. Today could have been an ordinary morning. The city could have woken up slowly. The men could have gone to work in the same grumbling way as yesterday. The trolleys could have arrived on time and sometimes a little late. A child could have had breakfast while a landlord and a tenant argued over rent. The men could have worked a full honest day, returned home, kissed their wives, and the city could have gone to bed. It could have just been Wednesday. But fates and other powers declared that this was not an ordinary morning.

Intro Cutscene to Day 2

Narrator:

From the chaos of the first day came the metered panic of the second. The fire spread as fire does, with apathetic disregard for the accomplishments of man. The already battered wooden homes were consumed and memories of them reduced to air. Marble baked and burned away into chalk. As their world was slowly vanishing into smoke, the people, as always, did what they could.

Final Cutscene: Victory

Life moves, but we must not let it sweep us away. We must learn to move over it, to control what little we can. It's what separates us from the animals.

Final Cutscene: Failure

Sometimes we are not prepared for our adversity. Sometimes the world does not give us an easy way. There are days when the world itself knocks the world down.

Appendix C Art Asset List

1. Models

a. Male1

i. Anims

1. walk,
2. run
3. pointing
4. digging
5. waving
6. crying
7. sitting
8. distraught/in shock
9. wounded/dying

•

b. Female1

i. Anims

1. walk,
2. run
3. pointing
4. digging
5. waving
6. crying
7. sitting
8. distraught/in shock
9. wounded/dying

•

c. House1

i. Collapsing (anim)s

d. House2

e. House3

f. House4

g. House5

h. Shelters

•

i. Props

i. Trolley and Trolley Rails

ii. Dead Horse

iii. Carriage / Cart

iv. Heirlooms

1. Watch,

- 2. Grandfather Clock,
- 3. Doll
- 4. Gas Can
- v. Debris (Interactable)
 - 1. 3 Different Stages / Meshes
- vi. Debris (Uninteractable Debris)
 - 1. Brick1
 - 2. Brick2
 - 3. Brick3
 - 4. Brick4
 - 5. Wood1 - > 5
 - 6. Girder
 - 7. Flaming Ash
 - 8. Wrecked Furniture (remains of chairs, tables, etc, for use in collapsed houses)
 - 9. Sign posts

•

- j. Hands (First Person)
 - i. Digging
 - ii. Walking
 - iii. Grabbing?
 - iv. Reaction to Fire

•

2. Cinematics

a. Opening

- i. Table/Teacup/Newspaper/Drapes/Window/Skyboxes

- b.** End of Day 1: Good
- c.** End of Day 1: Bad

d. End of Day 2: Good

e. End of Day 2: Bad

f. Ending

- i. Bad - Train

ii. Good - Wedding

3. **GUI**

- a. Health/Fire
- b. Status Bars Morale/Hope/Health
- c. Progress Gauge
- d. Reticules
- e. Time Element

4. **Other 2D Art**

- a. Decals
- b. Cracks, Proclamations

5. **Animations**

- a. Paper blowing through air / Down street

6. **Effects**

- a. Highlight Effect for Selectable Objects
- b. Dust
- c. Smoke
- d. Fire Particles/Glow
- e. There may be blood decals?

Appendix D Mechanic Description

1906 Mechanic Description:

09/06/10

Genre: First Person Survival

Outline: Enter the head of an anonymous citizen and try to survive an earthquake disaster over the course of three virtual days. Should you die you enter the body of another citizen (previously an NPC). The death of the previous body remains permanent.

Mechanic Technical Execution

Every object is given an AI-controller and a Player-controller script to determine its movement that can become both enabled or disabled. Normal NPCs, when spawned, will have only the AI controller enabled. When an NPC dies both AI-controller and Player-controller scripts are disabled and any further ragdoll physics is handled by the unity engine. When the player's NPC dies and enters a new NPC host, the AI-controller script of that NPC will be disabled and it's player-controller script will be enabled, allowing the player to take control of the next NPC almost seamlessly. Game logic for determining which NPC is chosen will be discussed later.

Movement

Basic first-person movement behavior seems to be the best option.

Running

Running would be a necessary, easy, and obvious aspect of movement control. No further discussion of it is necessary.

Encumbered

If the player grabs too many things, the ability to run is revoked and the walk speed is reduced. A static predefined variable and a boolean would be all that would be needed to be added to the movement logic.

Death Events

Instant Deaths

Collapsing buildings:

- for art assets an animation of a crumbling wall would be required, or a series of wall gibs generated from the original building mesh, sound effects of the collapsing building would be needed.
- in terms of technical execution only a collision check would be required for either case, in the case of prescribed events meshes would have to be replaced by gibs via scripting them in manually and only once per building asset through the use of prefabs. The crushed NPCs would only need to be ragdolled, a built in feature of unity.

- Death by Explosion:
 - Though almost all dynamite was handed by firemen or soldiers and the cause of very few of the actual casualties, it still would be necessary to implement.
 - Artistically gibs would have to be made for models similarly to the buildings.
 - Technically the gib placement would be a little more complicated, as it would have to be procedural every time. Parenting the gibs to the corresponding joints and assigning them a large initial force is the initial approach to the solution.

Shot for looting (potential):

- For art assets, shooting animation, as well as a gun and soldier model, would be required
- On the tech side a raycasting collision check would be involved, as well as a separate AI controller for the soldier entity. The player-controlled NPC would only need to be ragdolled.

-

Gradual Deaths

Fire:

- for art assets, fire and smoke particle textures would need to be made, as well as edited textures for whatever objects would be on fire and burnt. For the player a minimal fire effect would need to be created for the HUD.
- for tech, particles are already completely handled by unity, a status variable and a timer would be required for a person on fire, plus additional logic for AI of NPCs on fire.

- Drowning (potential):

- for art assets, some form of waterfront level would need to be made, including an animated water asset comparable to most professional games. A water/air notification would have to be built into the HUD as well.
- for tech, a collision check and a separate air timer would be needed.

Process for Determining Next Body

Control should not be given to the player when determining a new NPC host, as it is more technically difficult, will complicate the process of reviving, and may lead the player to

game the system by repeatedly killing and reviving in order to accomplish a goal or break the game. Only a few simple checks are required to automate the NPC selection process, which can be handled by the NPC AI script; is the NPC far enough away? did the NPC kill the player? is the NPC currently engaged in a task, hurt, or dead? These are simple boolean variables that would exist in the AI logic. Also, since all NPCs will be prefabs in unity, they are accessible by tags, which means we can iterate through them rather than pick randomly or build a list of them on initialization of the level.

“Hope” “Meter”

The game requires a feedback mechanism, a direct and almost immediate response to the players actions that determines by some degree what effect they are having.

We do not want it to be just hope. There always is some ambiguity in the result of emotional synthesis (especially during a large tragedy), many powerful emotions are being combined spontaneously. We also do not want it to be just a colored bar that fluctuates in width. Excessive meters, numbers, and general HUD elements detract from overall immersion.

It needs to be interpretable as a meter would. Ideally it would be represented with in-game behavior, such as ambient sound, an edited variable for a texture overlay, or NPC crowd behavior, but scope might need to be lowered. A minimal HUD element and a simple hope variable may be all that is possible.

Mechanic Options:

We aren't sure yet which mechanic is better, prototyping and playtesting may show us which fits the game the best.

- **Type 1: Personal Hope Meter & Health Indicator:**
 - Think only you matter? Try the personal hope meter, only certain events will affect you negatively (getting too close to a fire or collapsing building, witnessing death of an NPC) or positively (you retrieve an emotionally significant family heirloom, save a life, or find help)
- **Type 2: Community Morale Meter & Personal Health Indicator:**
 - The community is in trouble. Events such as total casualties and destruction will be the prime factors in calculating the variable. Personal well-being is also a factor, as you still can die.
- **Type 3: Both Community Morale and Personal Hope Meter:**
 - A combination of Types 1 and 2. The game may be cluttered or confusing due to the two different minimal HUD elements that would have to be interpreted separately, but doing so successfully would give us the best resulting game.

The game must above all be self-contained, believable, and engaging. Bonus points if we keep it morally ambiguous.

Progression Events

The sequence of events of the game will not be changed by the player, they may only be expedited. The end of each day has only two outcomes:

Victory - The player remains alive or dies rarely, and/or the player commits a majority of morally victorious actions (high hope or community morale).

Failure - The player dies often, or the player commits a majority of morally failing actions (low hope or community morale).

The outcome of either event leads to a positive or negative ending of each day, including a unique narration or cutscene for that outcome, as well as a difference in hope/morale meter value on the following day.

The major developments of each day remain unchanged, as it is a historic event. Most, if not all, of the events of the 1906 earthquake are horrible, and to say that they aren't, one would think, would be insulting. The endings would simply display a differing level of optimism.

Major endings.

Based on whether the majority of days are failures or victories, two separate endings are given. Likely just different cutscenes or final monologue, due to constraints. Could potentially be more complicated, such as in-game NPC dialogue.

The victorious or failing actions are eventually trade-offs, all are intended to both help and hinder you. You will have a harder time staying alive than you will questioning each action.

Actions:

Due to scope constraints, the actions of the player are categorized by necessity. Some are those that we need to have, some are things that we want to have, and some are things that would be really great if we had but probably won't get to.

- (Need)
 - Acquire personal items
 - Keeping your belongings with you will raise base morale and/or your hope meter. It will also hinder your movement.
 - Running
 - It's not a coward's thing to do if everyone does it. Save yourself first after all. May lower morale based on NPC complexity.
 - Extinguishing
 - Trying to stop the fire puts stress and risk upon yourself, but helps the community. Should you die while extinguishing a fire your charred body will likely not improve the situation.
 - Burn
 - Arson of your own property (when you have fire insurance) provides you personally with hope, but reduces the morale of the community should the fire keep catching.
- (Want)
 - Loot

- The owners are probably dead. You have nothing, is it theft if you take it from a dead man? Is it breaking the law if you don't get caught? May raise your hope meter, lower the morale, and give you certain advantages.
 - Rescuing
 - Helping other people often puts yourself in danger, but if you succeed I'm sure you've gained a friend.
 - Detonating
 - Blow it up. Blow it all up. Detonating allows you to clear away certain areas, potentially improving your hope, but may lower the morale of the community.
 - Technically challenging, but possible due to the built in explosion framework extension available in unity. AI would have to become more complicated if it were moved beyond prescribed events.
 - Artists would have to gib every building, possibly piggybacking the assets from the collapsed building gibs, as well as have new animations for laying and setting dynamite.
 - Crowd Directing:
 - See: http://web.archive.org/web/20071222203219/www.hlfallout.net/articles.php/hl2guide_51/2/ or http://en.wikipedia.org/wiki/Freedom_Fighters_%28video_game%29?
 - By far the most difficult this to do technically, and also the most difficult thing to do well mechanically.
 - Clearing Rubble
 - Digging away rubble to help someone will definitely improve the morale of the community, except if the person you rescue happens to be hysterical.
- (Like)
 - Steal/Rob (from an NPC)
 - For the type 1 mechanic it will boost your hope meter while attracting the attention of soldiers, for type 2 it will lower the morale of the citizens but give you an advantage based on the item you steal, and for type 3 it will boost your hope meter but lower the morale of others.

Further Shortening of the Scope

Instead of three days, only one day is implemented. Complete from beginning to end functionally, but lesser in overall narrative content and artistic workload.

An "infinite mode," where the player is constantly in a lethal environment is also an option, where there is no narrative content and even less art.

Appendix E: Calendar

1906 Calendar

The game experience is considered paramount throughout development for all decisions.

Term 1:

- **September:**
 - **18th: Weekly Saturday Meeting**
 - **Week of 19th:**
 - **Organization**
 - Work out how to implement story through-line dynamically
 - Integration of current assets
 - Write out context for Task list
 - Two demographics (male/female)
 - Three contextualizations each
 - **Tech**
 - Item pickup (raycasting)
 - EventsManager
 - **Art**
 - Model Buildings
 - Animations
 - Model Misc. Objects
 -
- **25th: Weekly Saturday Meeting**
- **Week of 26th:**
 - **Tech**
 - Procedural Animation
 - Integrate current walk animation & model
 - Randomized Appearance
 - Dynamic dying $\frac{1}{2}$
 - **Art**
 - Environment
 - Rubble
 - Animations
 - Scratch (upper body)
 - Shift (lower body)
 - Determine between Machinima/Rendered cutscenes
 - Cutscenes
 -

- **October:**
 - **2nd: Weekly Saturday Meeting**
 - make sure everyone knows the game
 -
 - **Week of 3rd: PRESENTATION**
 - **Tech**
 - Clearing Rubble
 - Cutsscenes ↵
 - Queueing Sound
 - Finish Dynamic Dying
 - Start AI
 - Environment
 - Terrain - Matt
 - Road
 - Fissures
 - Hero Building - Josh
 - Polish Existing Guy - Matt
 - Finish Up Girl - Billy
 - stick it in the body
 -
 -
 - **9th: Weekly Saturday Meeting**
 - **Week of 10th:**
 - **Tech**
 - Early AI
 - Random walk
 - Navmeshes
 - NPC spawning
 - Mechanic Tool
- Art**
- Alternate Guy - Matt (optional)
 - Shoes - Matt (optional)
 - Hair(s) - Billy
 - GUI art - Andrew
 - First Person Arm Animations - Billy
 - Trembling
 - Reach
 - Machinima/Rendered cutscenes
 - LOLNOPE

By the end of Term 1

Art - Rough versions of every asset in-game.

Tech - Functionality of every mechanic put in-game, at least semi-operational.

Term 2:

- **October:**
 - **30th: Weekly Saturday Meeting**
 - **Week of 31st:**
 - **Organization**
 -
 -
 -
 -
 -
 -
- **November:**
 - **6th: Weekly Saturday Meeting**
 - **Week of 7th:**
 - **Organization**
 -
 -
 -
 -
 - **13th: Weekly Saturday Meeting**
 - **Week of 14th:**
 - **Organization**
 -
 -
 -
 -
 - **20th: Weekly Saturday Meeting**

- **Week of 21st:**
 - **Organization**
 -
- Tech**
 -
- Art**
 -
- **27th: Weekly Saturday Meeting**
- **Week of 28th:**
 - **Tech**
 -
- Art**
 -
 - Begin making optional assets if things are going well
 -
 -
- **December:**
 - **4th: Weekly Saturday Meeting**
 - **Week of 5th: PRESENTATION**
 - **Tech**
 -
 - Art**
 - Art pass/optional assets
 -
 - **11th: Weekly Saturday Meeting**
 - **Week of 12th:**
 - **Tech**
 -
 - Art**
 - Art pass/optional assets

By the end of Term 2

Art - Good/Final versions of previous assets finished. NO NEW CONTENT.

Tech - Game fully functions, though not the most efficiently. Every major flaw in the code eradicated.

Term 3:

- **January:**
 - **15th: Weekly Saturday Meeting**
 - **Week of 16th:**
 - Start Writing MQP report
 - Begin recording with Voice Actors
 -
 - **22th: Weekly Saturday Meeting**
 - **Week of 23rd:**
 - Continue recording with VA
 -
 - **29th: Weekly Saturday Meeting**
 - **Week of 30th:**
 - Finalize recordings
 - Integrate sounds with events
 -
 -
- **February:**
 - **5th: Weekly Saturday Meeting**
 - **Week of 6th:**
 - Playtesting/Tweaking
 -
 - **12th: Weekly Saturday Meeting**
 - **Week of 13th:**
 - Last Tech Sweep
 - Last Art Integration
 -
 - **19th: Weekly Saturday Meeting**
 - **Week of 20th: PRESENTATION**
 - Continue to write MQP Report
 -
 - **26th: Weekly Saturday Meeting**
 - **Week of 27th:**

- Finish writing MQP Report
-
-
- **March:**
 - **4th: Last day of C term, Last day of MQP**
 - Submit Master Copies & Report
 -
 - **By the end of Term 3**

Art - Polished assets, optional pieces now can be made. Music and timing of events.
Tech - Post & Polish. Make code run faster/better.
Overall - Final assets and game submitted

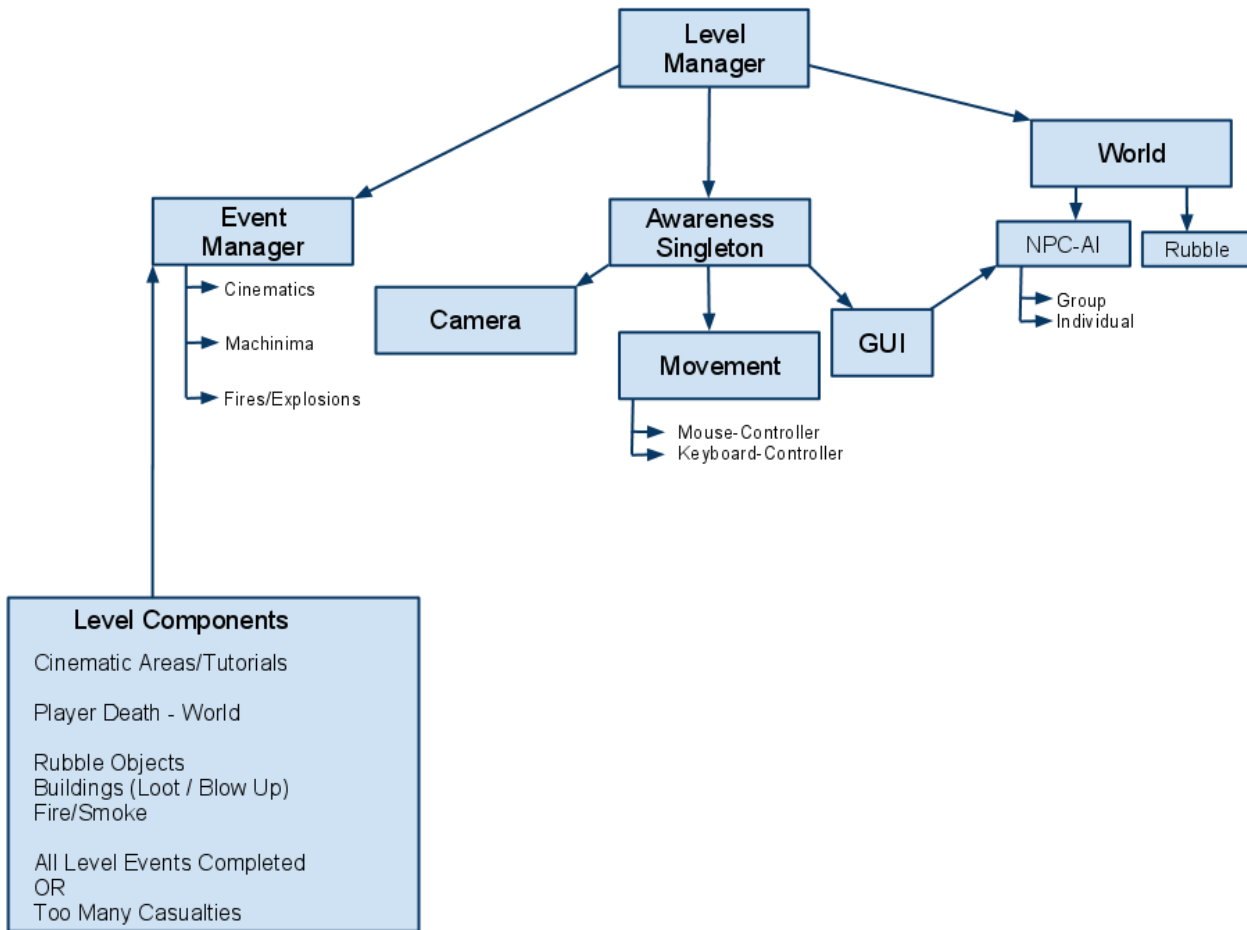
Appendix F: Tech

TECHNICAL EXECUTION:

Behavior to Tech Asset Table

Behavior	Tech Asset	
Camera Control	Camera Singleton	
Movement Control	Movement Singleton	Mouse-Controller Singleton Keyboard-Controller Singleton
GUI-State	GUI-Display Singleton	
Camera Control-Control Movement Control-Control GUI-State-Control	Awareness Singleton	
Randomized Appearance	Color Modification/Model Parenting	
Cinematics Machinima		
Level Change	Level Manager	
Fires/Explosions	Particle emitter aspect of the game object	
Rubble	Rubble object	replaceMesh, might be too much to have it all Physics Objects
NPC-AI Individual AI Group AI	Pathfinding Behavior Trees/Events Animation/Event Handling	

Dependency Tree



Appendix G: Story Throughline

Story Throughline (No death)

Day 1:

Earthquake event. Cutscene, then noon.

Crowd is in a blocked off neighborhood, people are unloading their belongings from their ruined houses out onto the street before the fire reaches them.

Finder Mission [Heirloom (Personal)]

Finder Mission [Heirloom (Neighbor)]

Some of the neighbors debate about the benefits of fire insurance.

Arson Mission (optional)

The crowd manages to make an opening in the rubble. The fire approaches, smoke drives everyone away. It is unsure if the houses survive.

As the crowd moves, people take what they can from their house.

Finder Mission [Heirloom (Personal)]

as well as their neighbors.

Steal Heirloom (Optional)

A small child cries out for a lost doll.

Finder Mission [Heirloom (Neighbor)] (Optional)

The day ends as the smoke rises, the cutscene summarizes the current state of the crowd.

Day 2:

Night. Fire. The crowd masses near the city square. With nothing to do they are poking to find anything else of interest or value.

Finder Mission [Relative(Personal)]

Steal Heirloom (Optional)

Suddenly explosions are everywhere, the army and fire brigade are dynamiting buildings for fire-block. For some ears simply ring, for others they are trapped beneath buildings, dead or simply doomed to be.

Finder Mission [Relative(Neighbor)](Optional)

The glow of the flames and the smoke in the air make it an evening to remember.

The day concludes in one of the Army camps set up for the citizens.

Visit Medic Tent (Optional)

Visit Food Line (Optional)

Day 3:

Noon. Conclusions.Success.

The crowds stand at the edge of a street, yards and yards of rubble stand between them and the last vestige of the fire.

Extinguish Fire(Optional)

Everyone is looking for someone or something, Everyone shouts names or digs with their hands in places they barely recognise but that they once called home. Looting is punishable by death by proclamation of the mayor, you wonder how many of these people are the actual owners of the things they're digging up.

Finder Mission [Relative(Neighbor)](Optional)

Finder Mission [Heirloom(Neighbor)](Optional)

Seal Heirloom (Optional)
Eventually the flames are beaten back and the smoke clears forever.

Appendix G: Task List

- **Death**
 - -Hope
 - -Morale
 - Health reset

Death is pretty straightforward. Dying in front of someone lowers them emotionally, and dying in front of a lot of people can lower their morale if all of you were trying to survive together.

- **Finder Missions**

The player can take it upon himself to raise his virtual spirits. Picking up an item, whomever it belongs to or for whatever reason. Through these basic actions the deed of looting can emerge rather than be inaccurately defined.

- Heirloom (Personal)
 - +Hope
 - -Speed
- Relative (Personal)
 - ++Hope
 - -Speed
- Heirloom (Neighbor)
 - +Morale
 - -Speed
- Relative (Neighbor)
 - ++Morale
 - -Speed
- Steal Heirloom
 - ---Morale
 - -Speed
 - +Hope

-

- **Clearing Debris**

The things worth having aren't above ground anymore. You must dig through the detritus that was once something if you want to find anything.

(2 to 15 Seconds)

Reveals an item on success, nothing on failure.

- **Dropping**

Dropping an item is the same as abandoning it. There usually isn't a penalty for doing that for most games, but this isn't a usual game. Abandoning someone is as significant an act as helping them.

- -Morale

- **Healing Missions**

- No one likes getting hurt. Luckily the kindness of strangers allow us a brief repose. The world won't stop for you, however, so some methods may have to be resorted to.
- Visit Medic Tent
 - (15 sec)
 - +Health
 - +Hope
- Steal Medkit
 - (2 sec)
 - ++Health
 - -Morale
 - (Risk) If seen by authority figures, you are reprimanded (--Hope/Death)
- Visit Food Line
 - (25 sec)
 - ++Health
- Steal Food
 - (2 sec)
 - ++Health
 - -Morale
 - (Risk) If seen by authority figures, you are reprimanded (--Hope/Death)

- **Arson**

We've all been there. That rare combination of events that make a man is debate the merits of burning all of his worldly possessions. Could be as simple and easy a task as pouring gasoline over a pile of junk for an approaching inferno, or the only slightly less easy task of dropping the match yourself. The neighbors probably won't be a fan of it.

- ++Hope
- -Morale
- (Risk) If seen by authority figures, you are reprimanded (--Hope/Death)

Appendix G: Barks

Man: Where is my horse?

Man: Oh Christ! Fire! Run!

Man: Watch out! It's collapsing!

Man: Our father in heaven... (lord's prayer)

Man: How could this happen? Things were supposed to be better here!

Man: Don't worry about me, just get the children out first!

Man: My foot is stuck! Someone help for the love of God!

Man: Things have really gone all higgledy-piggledy now

Man: Jane, can you hear me? Speak to me!

Man: Oh no, I see you eyeing that, but its mine, so don't go biting off more than you can chew!

Woman: Has anyone seen my husband?

Woman: Jeremy? Jeremy! Where is my child!

Woman: What have we done to deserve this?!

Woman: That's going to cost a pretty penny...

Woman: Of course I'm not alright! Are you off your trolley?!

Woman: Out of the fire and into the street is more like it!

Woman: Get off my back you mudraker!

Woman: Between you and me, if I Mayor ____ has really gotten my goat with this (decree?)

Woman: I told you we never should have come here, but does anyone listen to me?

Woman: Just praise the lord our house wasn't hit as bad as the Johnson's

Bibliography

"1906 San Francisco earthquake." *Wikipedia*. N.p., n.d. Web. 5 Oct 2010
<http://en.wikipedia.org/wiki/1906_San_Francisco_earthquake>.

Adobe Photoshop CS5. Adobe. Web. 2011.
<<http://www.adobe.com/products/photoshop/compare/>>.

Autodesk Maya. Autodesk, 2011. Web. 2011. <<http://usa.autodesk.com/maya/>>.

BioShock. 2K Games, 2007. Computer software.

"Eyewitnesses to the Earthquake and Fire." *The Virtual Museum of the City of San Francisco*. Web. 2011. <<http://www.sfmuseum.org/1906/ew.html>>.

“The Great 1906 San Francisco Earthquake.” *Earthquake Hazards Program*. N.p., n.d. 5 Oct 2010.
<<http://earthquake.usgs.gov/regional/nca/1906/18april/index.php>>.

Johansen, Emil. *Path*. Computer software. *Angry Ant*. Vers. 2. 15 Feb. 2011.
Web.<<http://eej.dk/angryant/>>.

Pixologic: Makers of ZBrush. 2011. Web. 2010. <<http://www.pixologic.com/home.php>>.

“Plenary Sessions.” *100th Anniversary 1906 San Francisco Earthquake Conference*. N.p., n.d.
Web. 5 Oct 2010
<<http://www.1906eqconf.org/plenarySessions.htm>>

“Quick Facts about the 1906 Earthquake and Fires.” *The Great 1906 Earthquakes & Fires of San Francisco*. N.p., n.d. Web. 5 Oct 2010.
<http://mceer.buffalo.edu/1906_Earthquake/san-francisco-earthquake.asp>

Red Dead Redemption. San Diego, California: Rockstar San Diego, 2010. Computer software.

"Unity 3." *Unity*. Unity Technologies. Web. 2011. <<http://unity3d.com/unity/>>.

Vijfwinkel, Marcel. *CGTextures*. Web. 2010. <<http://www.cgtextures.com/>>.

Citations

[1] "Eyewitnesses to the Earthquake and Fire." *The Virtual Museum of the City of San Francisco*. 2011. <http://www.sfmuseum.org/1906/ew.html>. (accessed February 12 2011)

[2] “The Great 1906 San Francisco Earthquake.” *Earthquake Hazards Program* <http://earthquake.usgs.gov/regional/nca/1906/18april/index.php> (accessed Oct 2010)

[3] “Plenary Sessions.” *100th Anniversary 1906 San Francisco Earthquake Conference* <http://www.1906eqconf.org/plenarySessions.htm>. (accessed 5 Oct 2010)

- [4] "Eyewitnesses to the Earthquake and Fire." *The Virtual Museum of the City of San Francisco*.<http://www.sfmuseum.org/1906/ew.html> (accessed Dec 22 2010)
- [5] "Quick Facts about the 1906 Earthquake and Fires." *The Great 1906 Earthquakes & Fires of San Francisco*.http://mceer.buffalo.edu/1906_Earthquake/san-francisco-earthquake.asp. (accessed Oct 27 2010)
- [6] "Quick Facts about the 1906 Earthquake and Fires." *The Great 1906 Earthquakes & Fires of San Francisco*.http://mceer.buffalo.edu/1906_Earthquake/san-francisco-earthquake.asp(accessed Oct 5 2010)
- [7] "The Great 1906 San Francisco Earthquake." *Earthquake Hazards Program*.<http://earthquake.usgs.gov/regional/nca/1906/18april/index.php>. (accessed Oct 5 2010)
- [8] "1906 San Francisco earthquake." *Wikipedia*.http://en.wikipedia.org/wiki/1906_San_Francisco_earthquake. (accessed Oct 5 2010)
- [9]*Red Dead Redemption*(Xbox 360 version) Rockstar San Diego (1998)
- [10] "Unity 3." *Unity*.Unity Technologies.<http://unity3d.com/unity/> (accessed Nov 11 2011)
- [11] Johansen, Emil. *Path*.Computer software.*Angry Ant*. Vers.<http://eej.dk/angryant/> (accessed Feb 15 2011)
- [12]*Autodesk Maya*. Autodesk.<http://usa.autodesk.com/maya/> (accessed January 10 2011)
- [13]*Adobe Photoshop CS5*. Adobe.<http://www.adobe.com/products/photoshop/compare> (accessed January 26 2011)
- [14] Vijfwinkel, Marcel. *CGTextures*.<http://www.cgtextures.com> (accessed Dec 2010)
- [15]*Zbrush*(version 4.0), Pixologic (2010)
- [16]*BioShock* (Xbox 360 version), 2K Games (2K Games, 2007)