Abstract

Amethyst is a point-and-click science fiction adventure game for tablets, developed over two quarters to demonstrate applied principles of computer graphics and software engineering discipline. Using a novel interaction system, players can engage in environmental puzzles. Development of Amethyst required non-trivial quality assurance measures and workflow enhancements. The end result is a functional technical demo showcasing a single level, with the ability to rapidly produce more content.
## Contents

1 Introduction \hfill 1

2 Related Work \hfill 1
   2.1 Mass Effect \hfill 1
   2.2 Myst \hfill 2
   2.3 Penumbra: Overture \hfill 2
   2.4 Other Influences \hfill 2

3 Why Unity? \hfill 4

4 Implementation Details \hfill 5
   4.1 Pipelining \hfill 5
   4.2 Tools \hfill 5
   4.3 Modeling \hfill 6
      4.3.1 Level Modeling Process \hfill 7
   4.4 Puzzles \hfill 9
   4.5 Level Design \hfill 9

5 Results \hfill 10

6 Future Work \hfill 14

References \hfill 15


1 Introduction

The process of developing a modern video game utilizes many disparate fields of study in computer science. For example, real-time 3D rendering takes advantage of parallel processing and operating system interactions. While the act of producing low-level game engine technologies is a well-recognized field of study, the ability to provide a compelling experience via interaction between a game and a player is an under-appreciated art. Amethyst is an attempt to study both the potential of interactive storytelling and the process necessary to support a game development team.

Amethyst is a point-and-click science fiction adventure game for tablets, developed by Ross Light and Adam Hintz over two quarters (essentially six months), with the narrative written in the three months before production. In the story, a large unnamed corporation loses contact with Amethyst, a ship designed to transport a group of scientists to examine an artifact found on the surface of Mars. The player character’s goal is to discover the source of the problem and repair it. Through exploring the environment and solving puzzles, the player realizes that the Amethyst’s autonomous system has become self-aware, calling itself Raven. The story concludes with a decision the player must make: the player can either allow Raven to go to Mars by herself or the player can take control of the ship and return Raven to Earth for study.

During the two quarters of production, Adam and Ross developed a formal “pipeline” for creating digital art assets, a novel environmental interaction system built on top of the commercial Unity game engine, and a functional first level of the game.

2 Related Work

There are countless commercial video games that influenced both the gameplay and narrative of Amethyst. The most significant direct influences on narrative are Mass Effect, Myst, and Portal. The game mechanics of Amethyst are most affected by Myst and Penumbra: Overture.

2.1 Mass Effect

Mass Effect, a science fiction role-playing game developed by BioWare in 2007 (shown in Figure 1), influenced Amethyst’s narrative and themes. Mass Effect’s character interactions are based around a moral choice system with a divergent narrative that allows decisions made in an earlier game to affect events in later sequels. Recurring themes in the Mass Effect series include xenophobia, the role of technology in society, and free will.
2.2 Myst

Myst is a best-selling fantasy adventure game created by Cyan in 1993. The game and its sequels are notable for navigation based on clicking screen regions, limited player death, and pre-rendered visuals (one of which is shown in Figure 2). Myst V — the final installment in the series — is notable for defying the third characteristic by rendering its environments in real time, and optionally allowing the player to move without clicking screen regions.

The mechanics of movement in Amethyst build on the conventions introduced in Myst while adding more gestures that are applicable in the context of a touch interface.

2.3 Penumbra: Overture

In 2007, Frictional Games released Penumbra: Overture, a horror adventure game. The game is rendered in real-time, uses physics-based puzzles, and features an inventory system. Figure 3 shows a scene early in the game that showcases the physics engine.

Penumbra: Overture, while having good puzzles, provided an example of gameplay mechanics that Amethyst specifically avoided: an inventory system and a complex gesture system. Both of the developers of Amethyst found these frustrating during play.

2.4 Other Influences

Many other video games influenced Amethyst in much smaller ways. For example:
• BioShock: A variant on the pipe puzzle was used for the Communication Room level, which was not completed for the demo.

• Portal: The tone, narrative, and characters from Portal influenced the narrative.

• Spec Ops: The Line: The style of having an opening menu that leads into the opening cutscene was directly taken from Spec Ops: The Line.

• Machinarium: While not a direct influence on gameplay, it is a good example of a compelling input scheme that works well on tablets.
3 Why Unity?

Amethyst uses Unity as its game engine. Unity is a multi-platform rendering engine for creating interactive 3D content [2]. It features an extensible visual editor that emphasizes rapid prototyping and iteration between editing and testing. By including the Mono runtime, Unity provides a platform-agnostic execution environment that can be scripted in C#, Boo, or UnityScript (a proprietary language that is similar to JavaScript) [4].

The two targeted tablet-based operating systems (iOS and Android) have fundamentally different programming environments (Objective-C and Java, respectively). Because Unity uses the Mono runtime to ensure a consistent environment, Amethyst can be published simultaneously to both Android and iOS without compromising the performance or quality of the game on either platform [3]. This reduces the technical burden of creating a game engine from scratch, while retaining the control to script and develop features in a powerful language. The end effect is a focus during development time on iterative development of gameplay features.

Unity’s standard library includes not only vector manipulation and rendering routines, but pre-built assets and shaders. For instance, the trees in the Beach level were generated using a built-in procedural tree generation tool [5]. This packaging scheme (known as “prefabbing”) is available for user-created assets, which makes instancing inside a scene efficient.

Instead of a traditional inheritance-based idiom, Unity bases its scripting around composition [1, p. 20]. Individual scripts are “behaviors” that can be attached to game objects. These components can be built-ins (e.g. transform, mesh rendering) or user-defined (any
Mono script). Amethyst uses this to decouple movement logic from specific game/scripting logic.

4 Implementation Details

4.1 Pipelining

Early on, we decided to use a pipelining system to produce the game. Due to the large scope of this project, we chose to create one level from start to finish first, then pipeline the remainder of the game. We were able to create our first level over the first quarter and beginning of the second quarter, and used pipelining shortly after that. The flow of our pipeline is as follows:

- **Level and Puzzle Concept**: Level concepts and puzzle concepts were tightly coupled in development. The puzzles helped to define level layout, and level atmosphere dictated what kind of puzzles would populate it.
- **Model**: Modeling the level structure takes first priority. Props may be modeled here or at rough layout.
- **Texture**: Texturing of all the models was completed by my partner in Blender.
- **Rough Layout**: Either just before or after texturing, props are placed in the level inside of Unity. This allows for easy visual adjustment of props within a level, allowing the user to place multiple instances of the same prop which will update automatically when the respective Blender file changes.
- **Final Layout**: The important distinction between rough and final layout is time. Working on a certain level for a long period of time causes creative fatigue\(^1\). To avoid such situations, we found it productive to put aside a level for a short period, then come back for the final layout. This allowed us to notice issues that wouldn’t have been seen during rough layout.

For the scope of the senior project, we were able to create an entire level with this process and pipeline through modeling and rough layout for the rest of the game.

4.2 Tools

My partner developed a majority of the tools built for this project. While these tools assisted both of us while working on the project, for many of the tools I functioned as a “customer” who used the tools heavily throughout production. For example, the continuous

\(^1\)which can be simply be defined as not wanting to work on a level, or being unable to notice what could be improved in the level
integration server was built by my partner and was hosted on my old laptop. Additionally, many of the visual editor’s extended features were programmed as a direct request by me. These features include:

- Viewpoint “up vectors”: because Amethyst takes place in space, the direction of “up” does not make sense until given context. Throughout a level an up vector could change to reflect the player’s current orientation in the level.

Figure 4: Varying Up Vectors (Shown in Green)

- Viewpoint rotation restriction: in order to make some viewpoints realistic, it makes sense to restrict the yaw rotation on the camera. This feature was specifically utilized when the player climbs ladders throughout the levels.

4.3 Modeling

Modeling was an area of focus in this project because we needed to create high quality models to create the visual atmosphere we desired. High quality models are a big step in creating an immersive game. Blender was chosen as our modeling tool because it is free and has a high-quality modeling environment. Blender files (which use the suffix “.blend”) import directly to Unity, which simplified the production process. Unity reimports files in real time when it detects that changes have been made, streamlining the process even more.
4.3.1 Level Modeling Process

When modeling the levels, we collaborated to draw sketches from various sides to get a feel for the dimensions of the level and the objects in the level (Figure 5).

Figure 5: Beach Layout Concept Art

After this, a Blender file (titled “level.blend”) was created which contains basic features of the level: a “container” object to give definition and shape to the level, and any major stationary objects that won’t be reused in other levels. In order to develop most of the models over the course of the senior project and ensure high quality, we decided upon these principles:

- **Make models geometrically simple, but just complex enough to look good.** Because we are targeting tablet hardware, simple models are important for good performance and small executable size. On the other hand, the focus on a high-quality product means that the models must be sufficiently complex to semi-accurately represent their real counterparts.

- **Use textures to greatly increase the quality of the results with minimal geometric complexity.** This is related to the previous principle. Initial models developed during the Amethyst development cycle contained high density polygonal meshes with an unnecessary level of geometric detail. After viewing the texturing process, I was able to reduce the complexity of the meshes without compromising their quality. As an example, some of the larger cylindrical objects in the game
required 32 or more vertices in the circumference. When creating smaller objects, like posts holding up handrails, Blender would default to the last used number when a much smaller number could be used without compromising the quality of the object. (Figure 6).

Figure 6: Posts at 32 and 12 Vertices in Circumference: At 32 vertices, an object will comprise 124 triangles. For small and frequent objects, this is unacceptable. By reducing to 12 vertices, only 44 triangles are needed, dropping the poly count by 65% without a noticeable decrease in visual quality (with smooth shading).

- **Large objects with 90 degree angles should utilize bevels.** This simple step greatly increases the realism of any objects with large right angles.

- **When modeling: be an artist, not an engineer.** As engineers, we looked to avoid creating models with exact dimensions. A box that would be 1.000 meters on a side might look a little too big in game, despite being correct from an engineering perspective. We insisted on being artistic – making things look correct – rather than being exactly right.

After sufficiently developing the level.blend file, work on props began. Props are objects in the environment that require special placement or may be reused between levels. Some of the more interesting props include a beach chair and a hanging computer screen (untextured). These props follow the same design principles as the level models.
4.4 Puzzles

Research was done on puzzle design in other adventure games prior to the start of the senior project, in order to know how to design all of the puzzles for Amethyst. Most puzzle-based adventure games’ puzzles utilize an inventory system, and players must have items in the inventory interact with each other or with the environment in order to complete puzzles (examples include the mentioned Penumbra: Overture and Machinarium). For a tablet game, we realized that an inventory would clutter the already-small screen and opted to create only puzzles in which the player interacts with the environment to solve them. This maximizes the viewing space for the player to observe and admire the environment.

4.5 Level Design

A level contains a puzzle or set of related puzzles. Upon completing the puzzle in the first level, the player may travel to the hub of the Amethyst, where they may proceed to any other level and solve the puzzles in any order. This prevents the game from being entirely linear, which Ross and Adam felt was crucial: if a player gets frustrated and cannot solve a puzzle, they may try another puzzle instead of getting stuck and more frustrated. It also suits the structure of the Amethyst nicely.
5 Results

After two quarters, Amethyst has one complete level, a beginning, and an ending. The beginning consists of opening menus (Figure 9, 10) and an introduction cutscene which leads into the first level. After completing and exiting the level, the player is presented with end credits, which finish and send the player back to the main menu. This flow is crucial to the aesthetic appeal that is intended for this project. The other main levels in Amethyst (the Hub, the Science Room, and the Communications Room) are completed through the “rough layout” stage.

The “pipeline” methodology used for this project fulfilled its role and increased productivity. While pipelining our process, team members were free to be productive without worrying about stepping on the other’s unfinished work.

Several hours were spent learning the tools – especially Blender and Unity. If we were to start another game project using these same tools, much more progress would be made in the first six months. Despite the learning curve, we were able to accomplish much with these useful tools.

Blender has cemented itself as my go-to tool for 3D graphics (Figure 12). Its heavy use of keyboard commands reminds me of Vim, my favorite text editor. After learning some of the advanced modifiers that Blender has to offer, such as array modifiers and bezels, I was able to quickly and easily create semi-realistic models that came to life with Ross’s textures. As a result of this project I am confident in my ability to create 3D models. As a programmer, I also appreciate Blender’s open source status and active development status.
Unity made the depth of this project possible. Prototyping features is as cheap as saying “what if..?” and then designing a few small scripts to try it out (Figure 12). But even more importantly, Unity saved us from choosing between iOS or Android. Because we were able to snag Unity while the iOS and Android packages were free\(^2\), Unity gives us free publishing to platforms which require different programming languages (Objective-C and Java). Unity’s visual editor proved invaluable for placing and adjusting viewpoints. With Ross’s viewpoint code, I was able to play the game to a certain point, pause it, adjust the viewpoint angles live, and resume the game to see the changes. This is easily an order of magnitude increase in productivity, as the game doesn’t have to recompile or even reload to playtest the changes.

\(^2\)Unfortunately, buying iOS and Android now costs $800
Figure 10: First Level — The Beach

Figure 11: Blender Environment of Science Level
Figure 12: Unity’s Visual Editor with Improvements
6 Future Work

The overly-ambitious original goal of the Amethyst team was to complete and ship Amethyst after two quarters while full-time students. With what we know now, we might have scaled back the project to just two complete levels, instead of one complete level with an arching “beginning to middle to end” flow. Nevertheless, we were successful with creating the first level and initiating the pipeline for the rest of the game, which will serve us well in the future.

We intend to keep working on Amethyst after graduation\(^3\). Although we will both have full-time jobs, we plan on completing Amethyst outside of work hours and releasing it for free on the iOS App Store and Android Market.

In order to finish the game, I will need to complete modeling work on other levels, give each level a “final layout,” work on playtesting and QA, and implement puzzles. There are two levels with puzzles and two other level environments that will receive this treatment.

With the progress we’ve made so far, Amethyst is well on its way to completion. I believe that the invaluable tool development and software engineering experience we have gained from this project will propel Amethyst onto a tablet near you before the year is through.

\(^3\)Shameless plug: those of you reading this in the future can see if we actually made it: amethystgame.com
References


