Lumenox: Aaru's Awakening

Final Project

Autumn 2012

Burkni J. Óskarsson
Ingþór Hjálmar Hjálmarsson
Tyrfingur Sigurðsson
BSc Computer Science – School Of Computer Science

Instructor: Stefán Freyr Stefánsson
Examiner: Bigir Kaldal Kristmannsson
Partner: Lumenox ehf.

T-404-LOKA
School of Computer Science

Table of Contents

Introduction

Aaru's Awakening is a 2d action platformer for multiple platforms. The game utilizes the Unity3D game engine and the mechanics are written in C#, supported by a few javascripts and lower level code for shaders.
The purpose of the project was to develop a fully functional game with emphasize on fast and practical game development.
The project was managed using the scrum agile methodology and took 10 weekly sprints.
This report includes information on how the development process went, what the final product is, retrospectives and other info on what was done in project Aaru's Awakening.

The project is done in collaboration with Lumenox Games. The students are members and owners of Lumenox Games. Lumenox Games also includes two artists and a sound engineer that will supply assets. The project's overseeing professor is Stefán Stefánsson.

Artifacts

- Final report
  - Detailed information about progress and how the project went (this document)
- Project planning document
  - Sprints, stories, work journal
- Code
  - The code snippets used in the project
- Game design document
  - Detailed information about the design of Lumenox: Aaru‘s Awakening
- Game client
  - Runtime client, playable on PC and Mac, the end product

Vision Statement

We intend to create the best possible platformer we can possibly make, a platformer with inspirational gameplay, intuitive graphics and the most dynamic puzzles. We want to give the gamers the experience of failure and hard earned wins leaving a lasting impression.

Early project

The Aaru's Awakening Project is built on an earlier prototype of a similar game that the group had worked on before. That game, Relocator, had the core mechanic that Aaru's Awakening uses, shooting a projectile into the world that you can teleport on. We scrapped everything from Relocator since this time we were going to develop a sellable, retail product that had to be developed more sophisticated than the Relocator assets and code.
We started by doing two off sprint weeks where we spent our time on game design and made the scope of the project clear. During those weeks we got our game design document started, set an outline for scrum (creating backlogs, deciding sprint lengths, coding rules and so forth).
Since the project had to be developed fast we decided to finish only a short design phase and get the details of the design done along with the product development, since it was probably going to change a lot during development.

Organization

Project management method(Scrum/Kanban)

Scrum is used for managing programming team while art team uses kanban. The kanban part is out of the scope of this project since it was not involved with the school project, but the scrum master assisted and taught the usage of kanban to the art director.

We used sprint length of 1 week in order to stay more agile and since we were working full time on the project. We picked scrum since we knew how to use it and it's the most common project management method for games.

Meetings

Daily standup - Every morning, people report in less than a minute what they have been doing and what they plan on doing during the day. 15 min max.
Monday Sprint planning - Scrum master holds a 15 minute meeting with team and it’s discussed what needs to be implemented first. Product backlog updated.
Friday sprint post mortem - Discussion on what went right and what went wrong during the sprint and what we want to change. Features reported to product owner and confirmed that finished sprint items are done done.
Meetup with teaching assistant - Discuss and report project progress, bringing up important issues that need to be shared with the school and acquire assistance with harder problems.

Art team

During the development of the game and more specifically the demo, we have been co-operating with skilled artists to create the environment, animations, sounds and assets. The art team was responsible for providing assets while our part of the asset pipeline was to integrate them with the client and made them work within the game world and some part of the development time went into writing shaders, adding particle systems and tuning the assets to work in game. The art team:

Ágúst Kristinsson - Art director
Jonatan Brüsch - Lead animator
Friðfinnur Occulus - Sound engineer

Project progress

Working hours

We estimate around 120 hour capacity per week meanwhile the autumn school semester hasn‘t started. Then we estimate 90-100 hours per week. When the project finished we had spent around 1200 man hours on the project.

Sprint burndown charts explained

The orange line stands for the estimated burndown in the sprint. The yellow line shows the sprint progression.
The green bars show work hours for each day that were spent on the sprint tasks. The brown bars show work hours for each day that were spent on other tasks - unexpected, discussions/meetings or business related.
The x axis represents days in the sprint and y axis shows number of hours.

Sprint Progress

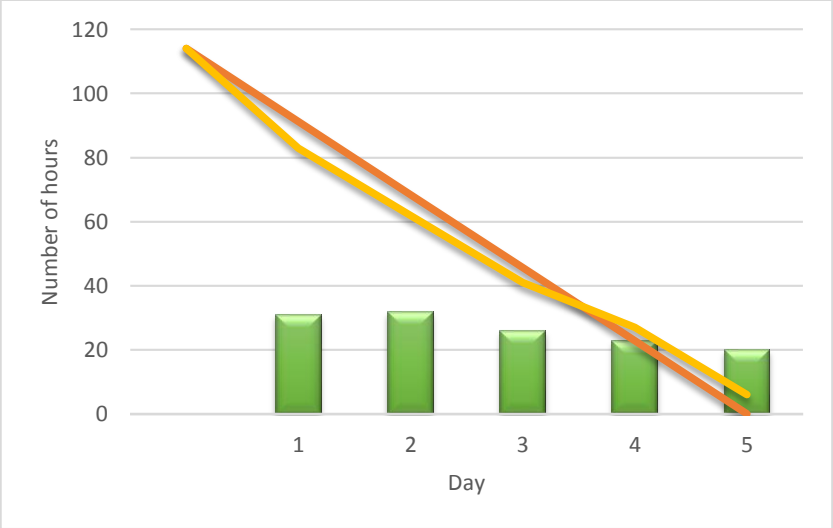
Sprint 1

Sprint overview:

This sprint was dedicated to the main character. Most of his attributes were started and finished in this sprint. The animation states went a long way and more than half of the main characters animations were put into the game.

Progress:

The sprint went quite well and 108/114 estimated hours were finished in 132 hours.



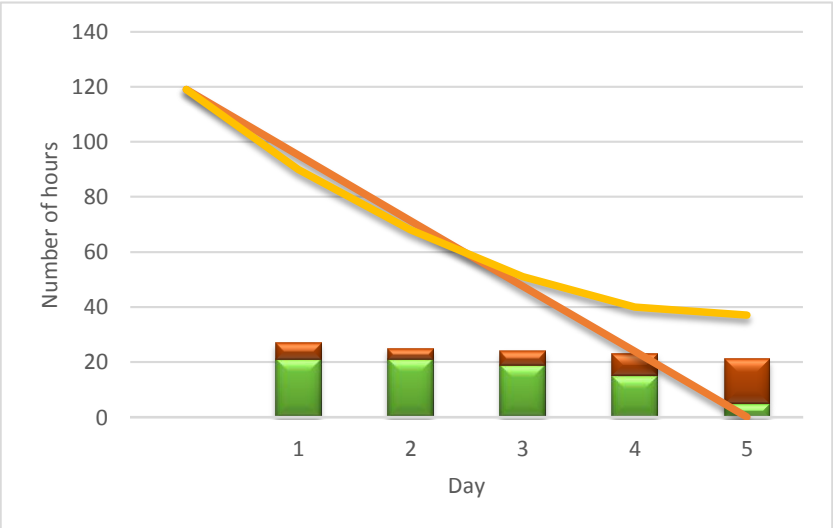
Sprint 2

Sprint overview:

This sprint wasn't dedicated to anything special. We went for things that were high in priority and on top of our minds. The first enemy, Eater, was implemented and took a lot less time than estimated. We started working on the Swarm, which turned out to be even harder implementing than we thought. We also started setting up the Dusk level.

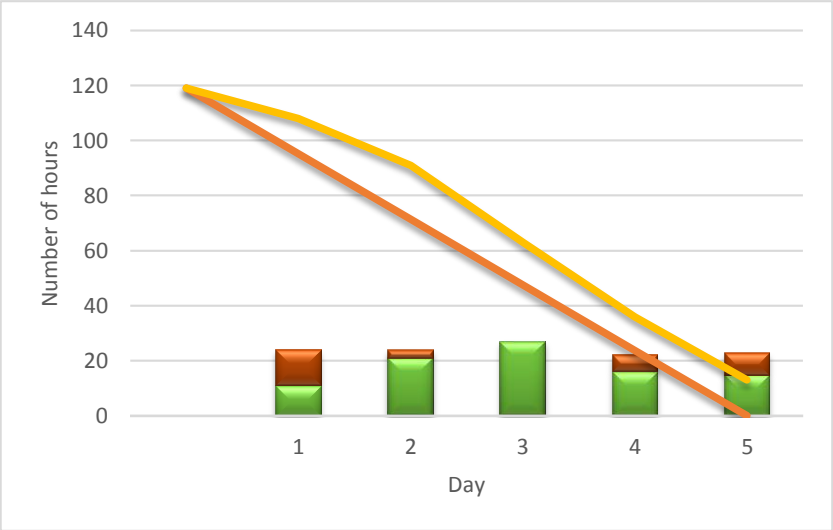
Progress:

The sprint was finished with 37 hours of tasks left, which is not good but the main reason for that was a lot of work needed on other unexpected tasks that were not in the sprint.



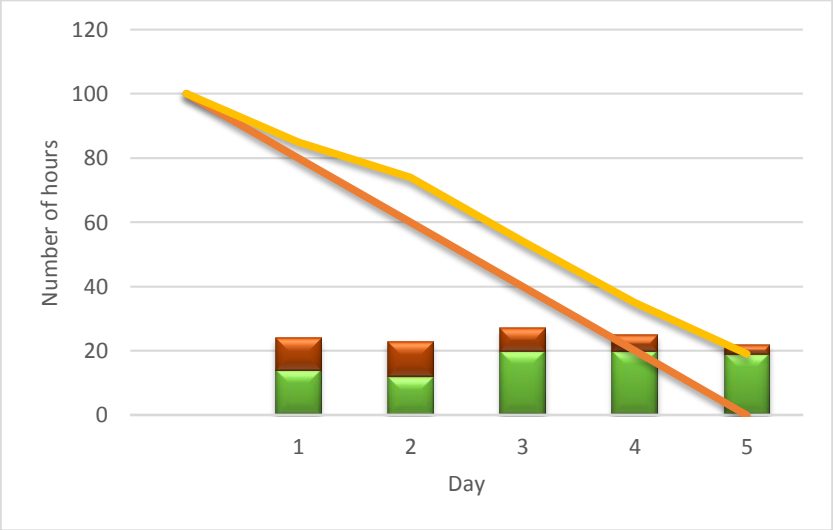
Sprint 3

Sprint overview:
In this sprint we wanted more life into the game so we decided to program and import some basic sound effects, along with animations for the teleport ability and placement of dot particles around the environment. The Swarm was continued but it was put on hold on day 2 because the method we were going for was not good enough. We started and completed setting up the tutorial level.
Progress:
This sprint went quite well if the Swarm is not taken into account - 13 hours left, with the 11 hours put on hold.



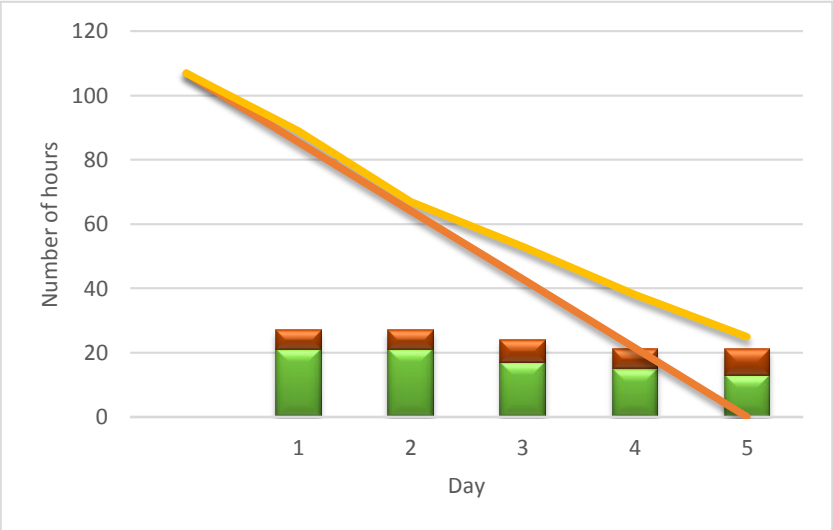
Sprint 4

Sprint overview:
The Dusk level setup was finished. Rest of the animation states were put down on this sprint and was nearly finished. The third enemy was implemented, Enemy Slimy - first one to be used in the game demo.
Progress:
Estimated hours 100 - Lower estimate than usual because of few meetings that were planned this week. The sprint was finished with 19 hours left, the other tasks took a bit more time than estimated.



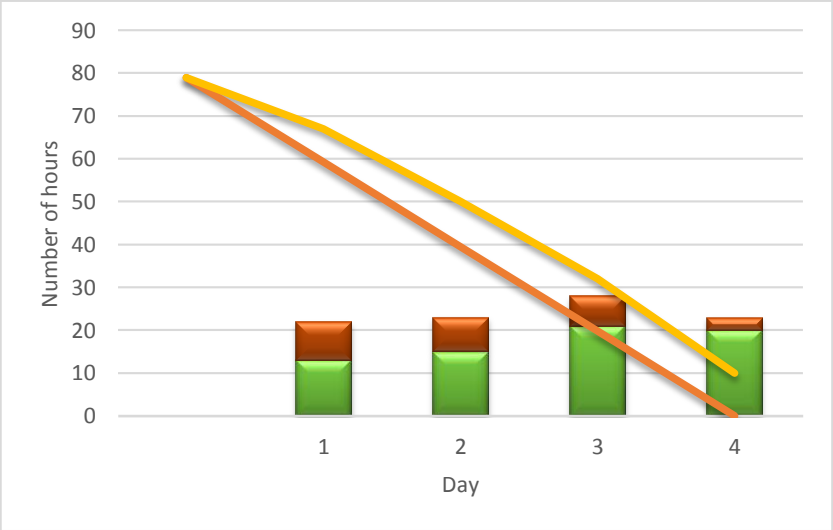
Sprint 5

Sprint overview:
A basic in-game menu was set up to play through the two levels. Game website was also set up. The fourth and fifth enemies were implemented, Enemy Flying Eater and Tongue - second and third to be used in the game demo. As we had already made few enemies we decided that we wanted player and enemy death right away and gave it some visual effects.
Progress:
At the end of this sprint we had 25 hours, 7 of which were canceled, left of the estimated work hours.



Sprint 6

Sprint overview:
The game website server was completed after some issues had been fixed with Apache and Tomcat colabriation. The visuals were improved by adding blinking eyes around the environment and slime effects on the deathzones. Public play testing was done and a few fixes were applied afterwards to the Dusk level based on the feedback we got.
Progress:
The sprint was finished with 10/80 estimated hours, with 70 hours going straight to the tasks.
This sprint only had 4 working days because of Lumenox Games summer vacation from 27.7-6.8.



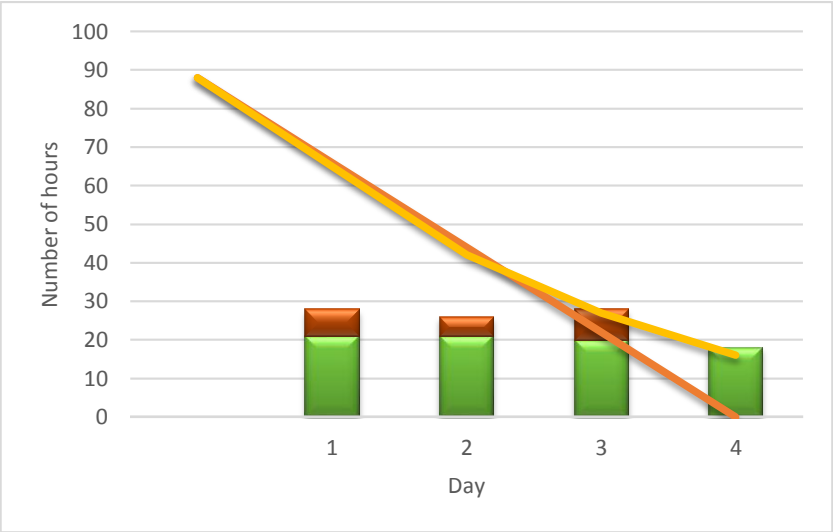
Sprint 7

Sprint overview:

We wanted to have some more interactivity in the game so we decided to add two big events to this sprint, the breaking bridge and slime event. Breaking bridge was finished but the slime event needed more optimization because of size. After last sprint public testing we saw that if the player was not a gamer they couldn't finish the game. So, we decided to have checkpoints in the game, but only when playing on easy difficulty. Another attempt on the Swarm was made, but again was not satisfying.

Progress:

The sprint was finished with 16/88 estimated hours left. Main reason for that is one of us was ill for a day. This sprint only had 4 working days because of Lumenox Games summer vacation from 27.7-6.8.



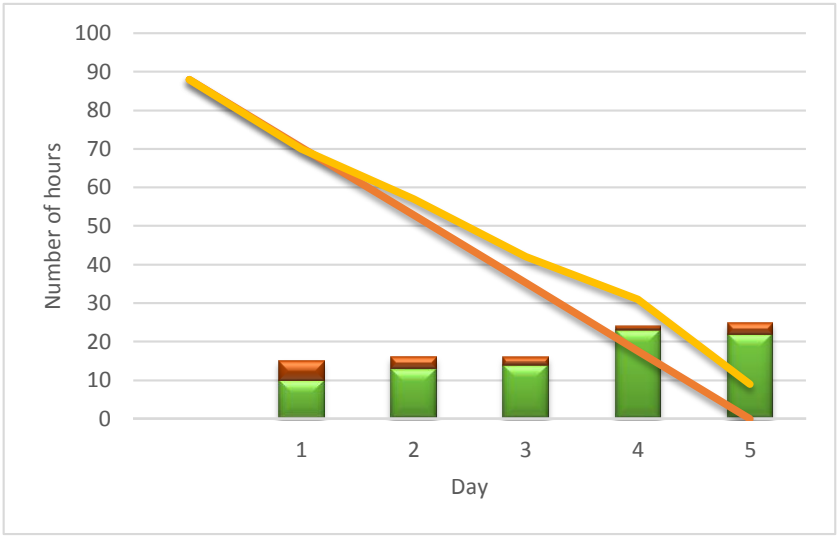
Sprint 8

Sprint overview:

This sprint was dedicated to optimizations and visual improvements. We started on creating the user interface in this sprint. We needed a startup screen, more beautiful main menu and an end of game screen. With the implementation of the main menu we did optimization for low end computers and added settings for the user to select the quality of the graphics before he started the game.

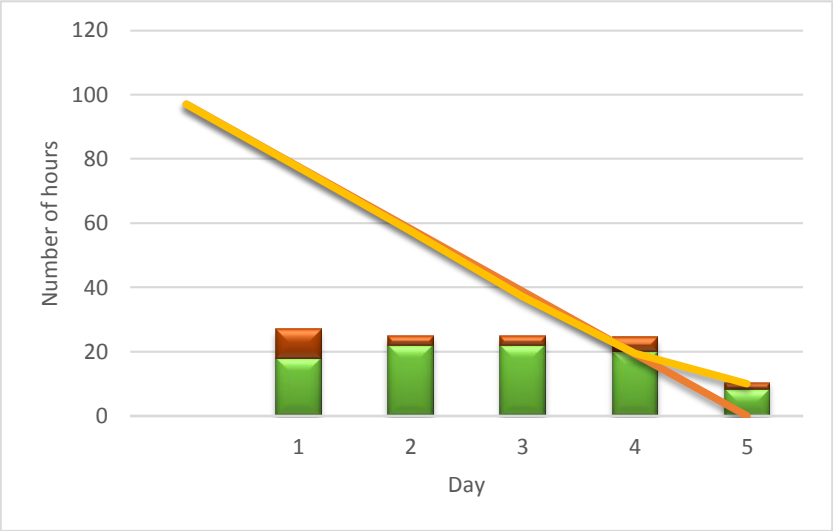
Progress:

The sprint was finished with 9 hours left of estimated time. One member was ill the first 3 days.


100
90
80
70
60
50
40
30
20
10
0
Number of hours
1
2
3
4
5
Day

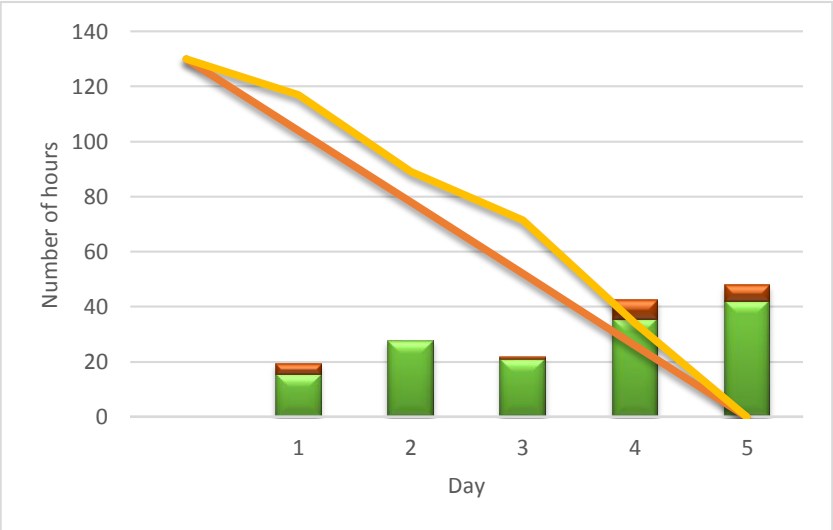
Sprint 9

Sprint overview:
Third and last attempt at the Swarm for the demo and again we were not satisfied. Among the things that were implemented were the light rays and 3d sounds. We had some fixes to do which were all in relation to the main character Aaru and his physics behavior in the environment.
Progress:
The sprint was finished with 10 hours left of estimated time. One member had to take the last day off



Sprint 10

Sprint overview:
Pickable items along with GUI were added here, which serve as the score mechanisim in the game. We also added the sound folies to the enemies and Aaru. At the end of this sprint we were going to publish the demo to Steam Greenlight, which is an approval system from Valve software to allow studios to get there games published on the Steam game platform. We completed our website and press release and did social networking for the release.
Progress:
We finished everything we had on this sprint, but to do that we had to spend a bit more time than estimated.



Usability testing

Usability testing was performed to spot and scale gameplay difficulty. We performed them on regular basis to measure if players could learn the game without outside information and to see if the player experience was enjoyable. The main purpose was measuring the fun factor and ease of access, although some bugs were spotted during usability testing.

Tutorial level testing

Observations were made on if the player received the hints on how to play the game correctly. Our requirement was that the player would find out how to control his avatar before becoming somewhat frustrated. He should also be able to finish the level.
At first we used discrete text telling the player what to do. The players did not notice so we changed the text's positioning and got somewhat better results. Later on we made the text more noisy and changed the text to be more intuitive. About 70% of players do notice the text if they don't find out the controls themselves. We were able to teach players how to move around, shoot and teleport and basic charging. The skill of in air charging was never learned through the tutorial, but it's not a required skill early in the game.

Dusk level testing

In the dusk level we were interested to see if the results from the tutorial level paid off and most of the time the player knew how to manipulate the character, but we have come to realise that the difficulty of the dusk level was too hard and the learning curve in the demo was too steep, and in our final release we will take more careful measures on the learning curve. Players also pointed out that animation frames could be higher for given characters in the dusk level, but they felt like the puzzles were cool. We also realized that we need to introduce new mechanics in a more simple manner with easy puzzles.

Tests performed

Public tests

Open demo test – Molinn

We performed an open demo test in Molinn at a ceremony where anyone could play the game when the tutorial was finished and the Dusk level was playable but not fully polished.

During the test, we observed the players without interrupting them and pointed down the things that could have been more enjoyable or if something did not work as intended. We also did informal interviews about what players liked and what they thought could be better. There was a large crowd at the ceremony, and anyone who wanted to try the game out was allowed to enter the test.

Guests at office

Kolfinna

19 years old female

No gaming experience

Tested if a person with no gaming experience could learn the controls and perform the puzzles.

Results: Could learn controls, could not solve puzzles that required fast cognitive reactions

Gústi‘s game designer friend

35 years old male

A professional game designer

Tested if a professional could understand the game, while he gave feedback on what he saw and thought

Results: He could play through all the puzzles on moderate speed

Feedback: The game mechanics were easy to understand, yet hard to master

Gústi‘s game designer friend‘s 12 years old boy

12 years old male

A mildly experienced gamer

Tested if a member of one of our largest market groups could learn to play the game

Results: Could play game just by finishing the tutorial level, he seemed addicted already

Gústi‘s game designer friend 8 years old boy

8 years old male

Highly interested gamer with low experience

Test if really young players can learn the game

Results: Took longer than average to learn the game, but after initial learning curve he could handle the game, although his cognitive reactions felt slower than in more experienced gamers, so an easy mode with no swarm(a mechanic that forces you to finish the level fast) would allow younger players to enjoy the game

Hjörtur Ólafsson

24 years old male

Competitive gaming experience

Test if the player would enjoy the high level experience of the game and hard modes

Results: Playing the game hard mode style was an enjoyable challenge that truly tested Hjörtur‘s gaming skills on a level he had not felt before in a single player game

Feedback: Put more moving objects in game, improove character controller

Hjörtur Guðmundsson

22 years old male

Competitive gaming experience

Test if the player would enjoy the high level experience of the game and hard modes

Results: He could finish the task but it was challenging to him

Feedback: Move charge button from E to space bar. We permanently changed the button and think that the feedback gave much value to the game.

Gísli Óskarsson

22 years old male

Gamer

Test if the typical gamer can finish all the puzzles

Results: Could not finish all puzzles

Feedback: Make learning curve less steep

Tools

Unity3D

Unity is our main tool to create Aaru's Awakening. It is a cross-platform game engine and IDE developed by Unity Technologies. With it we are able to port our game to platforms such as Windows, Macintosh, Linux, PlayStation, X-Box and Wii. The Unity IDE itself gives us the visual ability to arrange all the games assets in each level we create and keep them in a structured manner.

Component based engine

Unity3D is a component based engine which that the flow control and structure of the code is different than the standard in coding. Almost every entity in Unity has a main type GameObject. Everything else is of the type component. Components can be linked to GameObjects and GameObjects only. For example, a code snippet is a component and is applied to a GameObject. A 3D mesh layer is a component, and so is a sound and there is even a render component that tells the engine how to render the GameObject. Although this design may be confusing at first, it gets more comfortable and makes sense to developers as they get used to it. The biggest strengths of this architecture is simplicity and it makes it easy for GameObjects to talk to each other. You simply make a reference to the GameObject you want to talk to and perform actions on it's components, making fully compatible interaction between systems in the engine an easy task.

Monodevelop

Monodevelop is a programming IDE developed by Xamarin. Monodevelop is bundled with Unity for the development of the code for the assets in the Unity project. We use Monodevelop for all our C-Sharp code in Aaru's Awakening. Furthermore, Mono C# is a derivative of .net C# and has most of it's functionality. The one integrated into unity compiles into machine code, which makes unity games easy to write but still function in a fast manner.

Unity Asset server

The Unity Asset Server is an asset and version control system with a graphical user interface integrated into Unity. We keep our Unity project in the asset server. The Unity Asset server gives us the ability to rollback changes from previous revisions. It also is very crucial in terms of giving us a simple way to backup our project.

Asset pipeline

Adobe Photoshop

Adobe Photoshop is a graphics editing program developed and published by Adobe Systems. Our graphic designers use Photoshop as their main tool to create the rich graphics in Aaru's Awakening.

Adobe Premier

Adobe Premiere Pro is a timeline-based video editing software application developed and published by Adobe Systems. The graphic designers use Premiere to create cutscenes which will be in between levels.

TV Paint

TVPaint Animation is a 2D, bitmap-based digital animation software package, developed and distributed by TVPaint Developpement. The animators use TV Paint to do all the animations in Aaru's Awakening.

ImageMagick

Batch image manipulation tool. Programmers use for asset pipelining. We get the walk maps and other assets from artists and optimize them in imagemagick to use for the client. Since some older graphic cards and many graphic chipsets don't support texture sizes over 2048px, we use ImageMagick to optimize and cut images in smaller parts.

Unity Middleware

2Dtoolkit

A 2d sprite managament software. We import images that have been modified via imagemagick into the unity engine with 2d toolkit. 2dToolkit automatically creates sprite sheets and makes sure less space and performance is wasted by producing too many draw calls or storing empty data. Also provides an API interface to play animations and render 2d images within the Unity engine. Provides an interface to draw mesh colliders on 2d images.

AudioToolkit

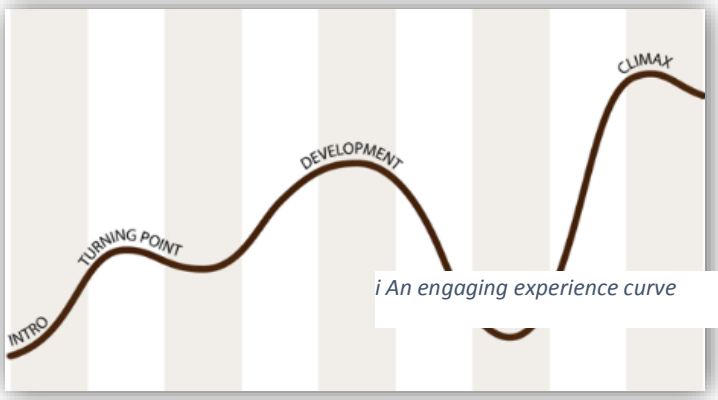
We felt the audio management could be made more intuitive and faster to use than the original Unity3d one. We tried out audio toolkit and it makes integrating audio to the game very easy and provides a more advanced audio API than the original Unity3d engine does.

iTween

A tweening middleware, a visual framework to create pre made in game movement and animations faster than with the original unity engine. Also provides a more advanced API to manage in game movement.

Leveldesign

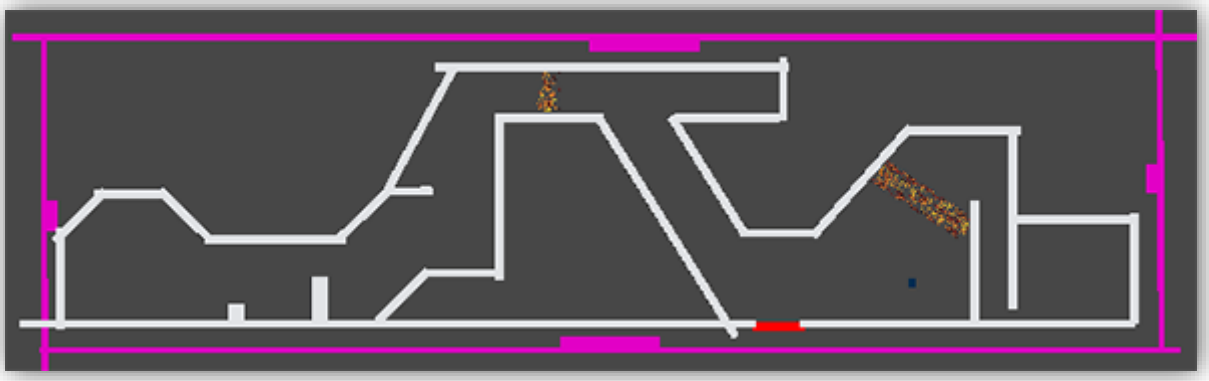
When designing levels we start out by drawing a level on paper with a mechanic sheet by our sides. We know what mechanics we can put and we think of the experience we want the user to have within the level. Often times we designed single puzzles first and tied them together into a level with an interesting experience curve



i An engaging experience curve

iAn engaging experience curve

With the design prototype we create a playable wireframe of the level and then test and adjust it until we are sure that is works as we expected.



ii Playable wireframe

When we are sure that the level works correctly, adds value to the game and is fun, we hand the wireframe over to the designers which paint the dreamy world we want Aaru's Awakening to be.

They start by sketching the look and feel of the level.

iii Design sketch

Once we are happy with the sketch, the designers go ahead and complete the final design of the level.

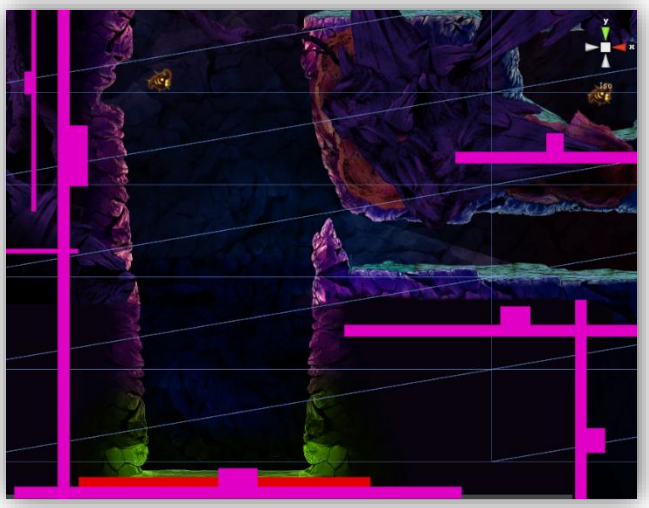
iv Final design

When the designers have completed there design, we start implementing the design with the game mechanics and make it playable.

Re-usable level modules

Camera colliders

Prefabs that decide whether if the camera can pass a specific zone - Lets us create less assets and control what the player can see at given moments.



vExample usage of camera colliders

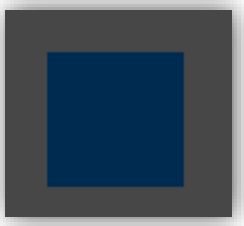
viThe red trigger is a death zone

Death zones

Areas that kill the player on touch - Most common way we use to kill the enemy, a trigger zone that sends a death message to the player and restarts the level if touched. The red lines are zones that can be scaled in any way you want and be parented to items. You can also assign the death zone script to any trigger.

Spawncubes

Position that spawns the player prefab at a specific point, also used to load variables used in every level, making it easier for us to set up new scenes



viiA Spawn cube, simple yet effective

Game mechanics and enemies

Mechanics, like spikes, enemies, and other objects are indeed developed to be usable again to make creating new levels easier. They can be placed by using the Unity scene viewer and set up only using that tool and the programming part has been abstracted. For example we use in game gizmos to set the waypoints instead of specifying coordinates in a program.

Problems of interest

The teleportation problem

The ball projectile is smaller than the player and can be at places where the player cannot be fit.
We saw multiple solutions for this problem
1. Give the player a prompt that he couldn't fit there with a sound and removing the ball
2. Killing the player because he didn't fit
3. Find an optimal space close that the player fits in
We made the decision on our priorities - Gameplay first, and we thought it was the most enjoyable to go for a hybrid between finding optimal space and killing the player. If it was an area where it's quite clear that there wasn't space for the player anywhere near, we'd kill the player. If it was something minimal where the player really felt like he could teleport there, we'd move him to the closest optimal space.

The Swarm

One of the game's features is a stress maker. A swarm that follows you through all the levels and if you're not fast enough, the swarm will reach your position and kill you, it's always behind you.
There were multiple requirements for the swarm,
1.it should be scaleable in size to fit the level's path,
2.It should move faster if the player was faster
3. It should be able to go into multiple paths
4. It should "feel" like it was a one big unit, yet still be independent (particles)
We made several attempts to get the swarm right, tried using a trigger system, tried animating it manually but as to this point we haven't finished it, but we think a variation of the Boids algorithm will solve the problem and plan on applying that solution next.

2d Parallax

To achieve the 2d Parallaxing, we use multiple cameras with different render layers. Suppose we have 4 different cameras, the main camera moves according the the player. The foreground camera renders the foreground on a layer on top of the main camera, and the background cameras render the layers in the back.
Then we move the cameras at different paces according to the player, and hence, we got 2d parallax mapping.

Enemies

The demo has three enemies. First of all we have the “Slimy”, which is a snail like creature which walks back and forth and leaves a toxic trail. The “Slimy” is sort of a neutral enemy because you actually need him, by jumping onto his shell you travel with him in the direction he is heading. But if you stay on him for too long and he turns to the other direction you will fall off his shell. On the other hand the “Slimy” leaves a toxic trail wherever he walks and this trail is lethal to the main character if he touches it.
The second enemy is the “Flying Eater”. He is more of and annoying enemy rather than dangerous. His main role in the game is to eat the players teleportation bullet, which in return renders the teleport ability of the main character useless. So when traveling through zones where the “Flying Eater” resides, the player needs to trust his ability of running, jumping and charging to get past the zone. The “Flying Eater” can be lethal but that is only if the main character goes in between the Eater and the teleportation bullet which he is trying to consume.
The third enemy is the "Tongue", which lashes out to eat bugs flying around the environment. If the player gets in between the tongue and the bugs, he gets knocked back.

Animation States of main character

The animation states became quite complex. We started out by creating a function that would monitor the input of the keyboard: “w” (jump), “a” moving left, “d” moving right, “space” special move and then the fall speed of the main character, whether he was grounded and if he was sliding. The combination of this information made us able to identify what character animation was supposed to be played at any given moment. The “space” key plays the special move which we call “Charge”, this ability makes the character charge into the direction of the pointer and overrides all other input. Apart from the special move the rest of the input works together to play the right animation. As of the demo release the main character has 20 different character animations when he is moving, when there is no input from the keyboard but the pointer is moving he will then always look at the pointer and then when there is no input from the keyboard and the pointer is not moving either then he has one idle animation.

End of school project

During the last weeks of our school project time we spent time to design an iteration on what we had already come up with and decided that we wanted to add a character progression system, animated cut-scenes and an achievement system to the game. As of this month, October 2012 we estimate that we have developed 25 levels with 4 boss encounters, 6 cut-scenes, a character development system and an achievement system by May 2012.
As for platforms, we aim to release on on PC, Linux, Mac, PS3, Wii and Xbox 360. With a first release on the desktop computers as soon as possible when development has finished.

Conclusion

After 10 weeks of work this summer, we had a working demo of Aaru's Awakening. The demo consists of two levels. A tutorial level to help players understand and learn how to control Aaru and then a fully functional level with puzzles and traps which gives a glimpse on how all the levels will be in the final release of the game. We would have liked to be able to use some of code from the original alpha build but realized that we would need to program the game from scratch, so we were only able to use logic on e.g. how traps would work and such. We did all the programming in CSharp, which was good and gave us better control of the game but more challenging was working with all the graphical elements in the game. As we were not used to working with so complex graphics and all the parallax the game has to offer, then setting up the levels and programming became quite hard, but we worked our way through that and learned a lot on the way.
We are very proud of the demo of Aaru's Awakening, it was very demanding and satisfying to complete it. It's our impression that our game will be well received by the gaming community and we have gotten very good feedback on the demo. Among all the feedback we have gotten, the one we are most proud of is from Nick Jovic, Regional Sales Director at Unity Technologies, he said to us that this was exactly what Unity needed to demonstrate what is possible to do in Unity, as their problem is that people assume that you can only do 3D games in Unity. He also further added that he would like to put the game up on the Unity site when its ready.

Summary

All in all we accomplish what we set out to do. The demo of Aaru's Awakening was all we expected it to be and more. From the beginning we very organized in the manner how we approached the project and we managed to tile everything up in the correct order of things, focused and knowing what we wanted to achieve. In retrospect we think that we wouldn't have wanted to do the project in any other manner as the result exceeded our expectation. We always knew that we would meet obstacles on our way but we were up to the challenge and we're not going to let anything stop us as it is our dream to see this game completed. This attitude helped us alot at times when we were in trouble and we always found solutions or ways to go about. We have learned a lot about how Unity works, programming in Unity, managing a team of designers but most of all we've learned teamwork and how teamwork makes things happen.
Ahead of us is long road of work so we can complete the full version of Aaru's Awakening, but this summers work has laid strong grounds to complete the task, and the team is stronger than ever before, ready to complete the mission.

Project Retrospective

What went well:

- Teamwork
- Game engine choice
- Asset pipeline
- The end product is like we wanted it to become
- Being an actual stakeholder in the project (increased motivation)
- Usage of middle-ware and an already field tested engine instead of making everything ourselves from scratch (do things like they are done in practice)

What we would change on next project

- Make more detailed designs before implementation
- Longer sprints(fewer meetings)
- Use a more sophisticated software than excel to manage sprints
- Make the tutorial level last
- Allocate tasks better per person
- Define a stronger project scope

What experience this project left us with

- Working in an environment where excellence is expected, on a serious product that's to be sold to customers
- Creating an industry competitive game from A-Z
- Expansive knowledge of using the Unity3D game engine (industry standard for coming years)
- Managing a full team of game developers
- More knowledge of real time applications and usage of forces in 3d space
- Slight introduction to shader programming
- Knowledge of component based programming
- Strengthened C# skills

LUMENOX

GAMES

AARU'S AWAKENING