Proceedings of The National Conference
On Undergraduate Research (NCUR) 2016
University of North Carolina Asheville
Asheville, North Carolina
April 7-9, 2016

# Albatross: A 2D Action Adventure Role Play Game with a Twist

Tyler Strickland
Computer Science
The University of North Carolina at Asheville
One University Heights
Asheville, NC, 28804, USA

Faculty Advisors: Dean Brock, Rebecca Bruce, Marietta Cameron, Susan Reiser, Kevin Sanft, Charles Shaeffer, Adam Whitley

## Abstract

*Albatross* was developed using the Unity 5 engine, and includes many elements of an Action Role Playing Game or ARPG, such as an experience system, equipment, fast battles, etc. However the point of the game is to not only complete the level in an action adventure fashion (similar to games like *Zelda II: The Adventure of Link* and the *Ys* series) but also to solve puzzles or challenges in order to advance past enemies and through the level. By destroying enemies haphazardly, the character's "burden meter" increases; this is represented by the albatross the character is forced to carry. If the albatross gets too heavy, it defeats the character. If the character finds other ways of defeating or getting around enemies, the albatross' weight does not increase. The burden meter is the "twist"; it adds a layer of complexity to the battle play. The goal of this project was to create a concept computer based demo for a console based ARPG. The game has three levels, and the levels take on average around four minutes to complete depending on which level the player is on and level complexity. The three levels will vary in enemy layout and monsters, and the final level will have a harder enemy to defeat. The project was iteratively tested and improved over two cycles by five users of varying technical skills.

**Keywords: Unity 5, Game Development, User Testing**

## 1. Introduction

*Albatross* was conceived by determining an interesting tactic for an Action Role Playing Game (ARPG). This idea was incorporated in this project because it was a way to make the game challenging and to be used as a plot point in the game's story. Essentially, the player cannot hurt the enemies due to personal conviction or moral code. Other games have used moral or religious code as a plot point to make the game challenging or to have a "good ending" to the story. However, since this game's story is yet to be determined, the challenge was applying this concept without a story to drive it. The purpose of this research was to implement a challenging game mechanic while exercising user testing to find the appropriate use for said game mechanic.

## 2. Usability and Its Importance in Game Development

In game development, there are many factors that one must consider before the game is distributed for players to enjoy. Usability is important to the developer as well as the user. Usability forces the developer to consider the perspective of the user, and that perspective is crucial to developing a successful product. As Laitinen pointed, usability provides useful insight into what a developer needs to add, subtract, or eliminate to make the game more

user-accessible to the desired demographic.[5] It ensures the user has the best possible experience, which is critical for all applications, not just games.

Laitinen shows the importance of user testing and usability. *Albatross* incorporates similar testing. Testing methods include selecting users from a wide demographic, as suggested by Browne, Annd, and Terry[1, 3,] among others. Their data suggest that assuring developers employ user testers of various skillsets helps elicit ideas for improvement and identify problems. Graham suggests including some quality assurance testing along with usability testing[4.] The combination of both styles of testing helps streamline the development process since it mixes UI repairing with bug repairing. Finally, when considering what to test, it is important to streamline one's categories, questions, and testing goals so the developer can get the most out of the tests, as stated by Hookham, et al[2.]

In summary, the process of UX testing is imperative to not just the gaming development community, but to development as a whole. User testing provides crucial information including the identification of problems and also provides the developer with feedback from their target demographic.

## 3. Hardware and Software Used

Unity is the engine chosen to develop this game. In the future, the plan is to port this game to home video game consoles, and Unity provides the most console support. The Unity engine innately supports cross platform support for games developed for Windows, OS X, and even Linux operating systems. Development was done on a MacBook running OS X 10.11 ("El Capitan"), so cross platform support is a benefit to development. The free version of Unity supports easy integration with several home video game consoles and mobile devices including Android, iOS, Microsoft Xbox 360, and Sony Playstation 3. In fact, Nintendo suggests developers use Unity for the Nintendo Wii U. Unity is a very versatile engine.

Unity supports two native languages: JavaScript (called UnityScript in Unity) and C#. C# is preferred over JavaScript because C# uses variable declarations, which makes organizing variables much easier. C# is also much cleaner than JavaScript and very similar in form to C++. This makes communicating to other developers more simple to translate to developers who may or may not know the C# language, especially since C++ is widely used in the video game development community. C# is also the preferred language for other engines such as the Unreal engine. In theory, this will facilitate the transition to another engine if need arises.

## 4. Development

Planning began in November 2015 with the desire to create a game that would attract players. Then, the idea of the burden system was created as a unique game twist. Brainstorming ensued for the next month. When development began in January, some very basic character movement was developed as part of a full demo of a game. This prototype would entice future developers, especially game artists to help create assets for the game. The game prototype will ideally evolve to a fully functional game that can be published for a general audience to enjoy. This section highlights the development process to date.

Since the author's previous experience with Unity is limited, the process of development has been an educational experience. The process could be categorized as rife with just-in-time learning and problem solving. Every feature that needed to be implemented required hours of reading the Unity API to understand the necessary methods, data, and recommended approaches. Some of the favorite achievements of the game are the falling spikes. The spikes need to descend upon the player located beneath them. Once the spikes hit the ground object, after a set time, the spikes return to their origin location. The key to this is a collision detector on the ground, which triggers the object to fall. This mechanic was designed very similar to the *Super Mario Bros.* games in that the objects fall very fast and then slowly creep back into position, *Albatross* achieves this goal, although it took several weeks to perfect.

The next highlighted achievement was the burden system. The system makes the player heavier if the player attacks the enemy directly. The point of this system is to find an alternative way to defeat the enemy using traps, pitfalls, or by beating the enemy to goal. The reward for defeating the enemy passively is the player receives more health point ups (HP), money, and experience (XP). These rewards are disbursed to the player in the background rather than falling on the ground for the player to collect. This assures the player does not need to risk a life to get the rewards if the player, for example, tricks the enemy into falling into a pit. If the player attacks directly, the game becomes significantly harder because the player becomes heavier. This is achieved by increasing the mass of the

player's 2DRigidbody component by 0.01 units of mass. To make the game playable, however, the burden lessens overtime. Selecting the "hard mode" in the game can activate the burden system.

Enemy AI was one of the more time consuming aspects of this project. It currently serves its purpose as the enemies run around, fly, or bounce to hurt the player. In addition, they shoot at the player in order to cause damage. Touching these enemies will also hurt the player. However, the AI is basic. Enemies are able to run back and forth and chase the player—sometimes—as well as fire at the player. However, if the invisible walking track is not placed correctly or the enemy runs off the track, the enemy sometimes gets stuck. A significant portion of development time has been devoted to fixing this issue.

## 5. User Testing

### 5.1 Method

At least five people will test *Albatross*. The game's executable will be known as the app file, and it will boot upon load. The users will receive an app file that is compatible with all three major operating systems: Windows, OS X, and Linux. Users varied in age, game skillsets, and interest because the future plans for this game is to distribute. The test plan consists of distributing the app files and a link to a Google Forms document to the testers. Google Forms allows for convenient, easy, and swift compilation of the user test data. The questions are intended to be both subjective and objective so answers can be specific but also allow users to speak their mind about the project. There is also a demographic section to correlate test data gamer demographics.

### 5.2 Results

Testing proved to be extremely helpful in the project. Users gave critical and pragmatic advice and identified flaws that were fixed and retested by new user testers. Fifteen testers agreed to play the game and give a fair analysis of the game's current state. Google Forms proved to be a formidable asset in user testing. All except one user were Windows users and all the users experienced no problems booting the game and playing. This is exciting given the fact the game was programed on a MacBook Pro. Figure 1 shows the OS users.

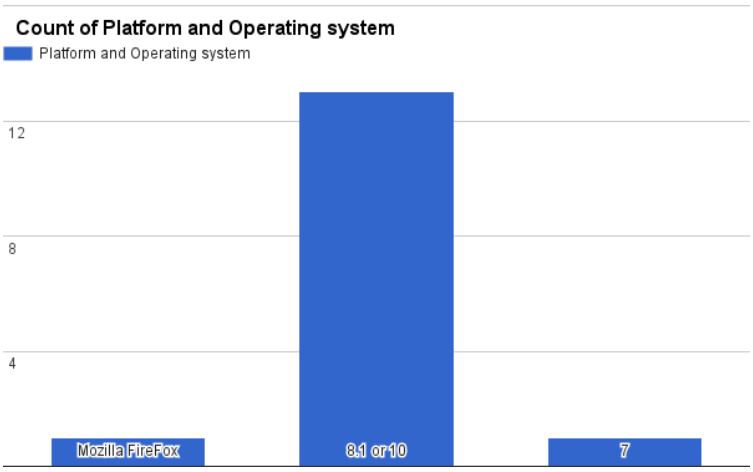


Figure 1. Operating systems used by each tester.

The one tester was a web browser user. Web browser support was added as a secondary option for users to test the game without downloading the app file. However, the web browser immediately proved to be extremely buggy with poor performance and other interesting occurrences that required immediate suspension of that port of the game. This was not surprising. The browser version was removed because it was not a target platform and additional development time repairing this port was not necessary. Performance tests passed on other platforms. There are three builds of *Albatross*: v0.5, v0.6, and v0.9.2. The game did experience some crashing with build v0.6. Of the fifteen times tested, the game crashed three times. Build 0.6 included music and the option of toggling the music on

and off. The game, as reported by one of these three users, "When I was messing with the menu the game would not respond. I had to access the task manager to exit the program."[6] This problem could not be replicated in both the Macintosh and Windows bug testing. Users also felt the music was not suitable and expressed distaste for the music. The music was removed in build 0.9.2 of the game. Figure 2 shows what users felt about the game overall in terms of enjoyment and level of challenge

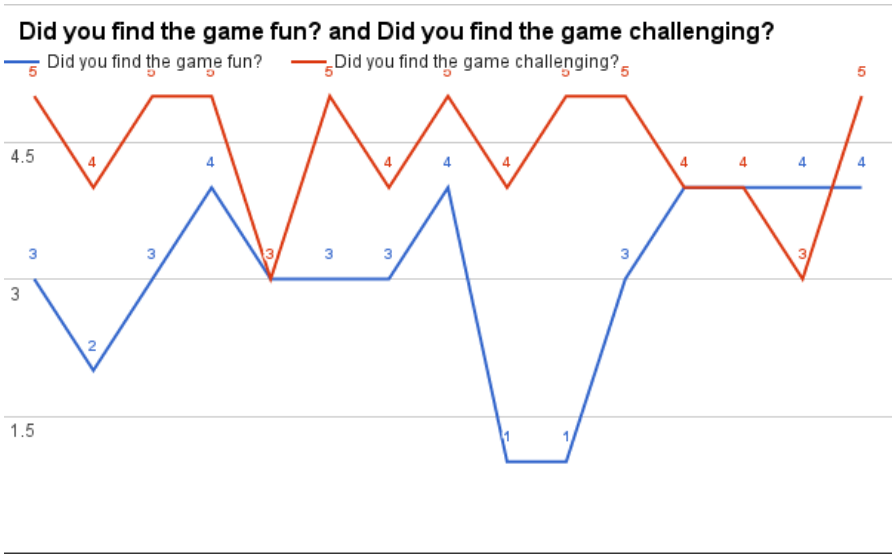


Figure 2. Users respond to the game's level of fun and level of challenge where 0 is the least and 5 is the most. Red represents challenge and blue represents fun.

The game proved to be fun but challenging. One user stated, "Found it to be challenging and a bit addictive...it was fun."[6] In build 0.5, instructions were not visible to the players. They did not know the game had an instruction menu at the start of the game. The users also reported no knowledge about various features in the game. For example, one user even reported not knowing about the fire button. There was an assumption that the amount of difficulty players were having was attributed to the lack of instructions given to the player. However, this later proved to be partially true as repeat testers were able to play the game but new players still quoted problems with the game's difficulty. In build 0.6 and subsequent builds, the instruction menu was arranged to be the first menu the player saw when clicking the "play game" button. This improved overall scores when asked about the clarity of instructions.

The users responded the Burden system was not something they would like to see in a game like this. In fact, when asked, thirty percent of the users answered, "No; the system is okay, but it's not something I would like to play in this kind of video game." Most users equated this opinion with the lack of abilities to defeat enemies passively. While there were several instances of passively defeating enemies, it seems that an even bigger emphasis of traps, hazards, and more is needed to make the Burden system relevant to the game, which requires more time into development. One user even alluded to the use of story to force the player into using the system, and, in fact, using a time-based system rather than an attack-based system. Users thought there needed to be an instance in which the player needs to complete the level in a short amount of time rather than forbidding the player to attack enemies. This would make a much more enjoyable experience for the Burden system. Future development will consider these responses. Figure 3 shows the age spread.

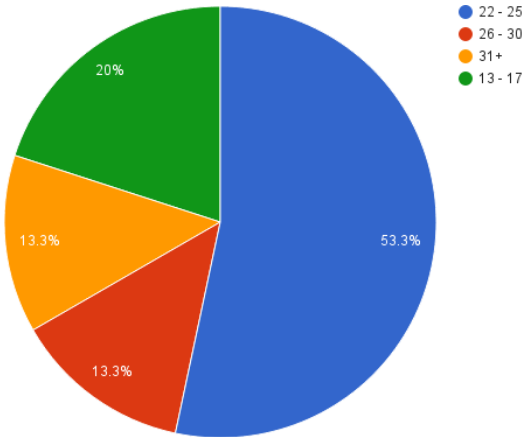


Figure 3. The age range of test users. The chart shows that the majority of users were, in fact, ages 22 – 25

The demographic was diverse. Since users were anonymous, a wide range of ages was represented. This shows how to develop the game in the future when story and visual style is considered.

Nearly sixty-seven percent of users play games frequently, at least 1 hour a day. Nearly all of the users prefer to play Action-Platformers, Action-Adventure, and Japanese based Role Playing, which includes many Action RPG genres as well as turn based RPG. These users happen to be the ideal type for a game like this because the game will fall under all three of these categories.

## 6. Conclusion and Future Work

### 6.1 Conclusion

Users proved to be a crucial part of the development of *Albatross.* The users showed that the burden game mechanic needed a story to drive it. The game mechanic seemed, to users, as out of place, but did achieve the goal of being a challenge to said users. The game had few operational bugs upon its current build among all machines tested. The users who tested *Albatross* were diverse. Disappointingly, I was unable to find a user with a Macintosh or Linux based machine; however, the users who tested displayed a broad spectrum of machine specifications. This is crucial when determining machine specification for possible commercial distribution.

### 6.2 Future work

*Albatross* is far from finished. Development must continue to give it a more polished state and to prepare for a public release. Planned modifications include: develop a story in which the burden system can thrive, add more features to the main character for customization (weapons, armor, etc.), more complex levels and hazards, and develop a robust enemy roster and AI. The game needs an artist that specializes in digital animation to create a world in which the game will exist. The artist will also need to be able to use Unity and adapt their work into the game's current code. Once that has been negotiated, the process of transferring the work to console is next. The game will need to be optimized for whichever video game company will allow the game to be developed for their console. Negotiations have begun with Nintendo. Sony will also be contacted. If the console companies deny access to development tools, *Albatross* will be released as a PC game. Upon successful crowd funding, the project's development will continue with the goal of being deployed to console and/or Steam markets.

## 7. References

1. Kevin Browne and Christopher Anand. 2013. Gamification and serious game approaches for introductory computer science tablet software. In Proceedings of the First International Conference on Gameful Design,

Research, and Applications (Gamification '13). ACM, New York, NY, USA, 50-57. DOI=http://0-dx.doi.org.wncln.wncln.org/10.1145/2583008.2583015

2. Geoffrey Hookham, Keith Nesbitt, and Frances Kay-Lambkin. 2016. Comparing usability and engagement between a serious game and a traditional online program. In Proceedings of the Australasian Computer Science Week Multiconference (ACSW '16). ACM, New York, NY, USA, , Article 54 , 10 pages. DOI=http://0-dx.doi.org.wncln.wncln.org/10.1145/2843043.2843365

3. Michael Terry, Matthew Kay, and Ben Lafreniere. 2010. Perceptions and practices of usability in the free/open source software (FoSS) community. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 999-1008. DOI=http://0-dx.doi.org.wncln.wncln.org/10.1145/1753326.1753476

4. Riley Graham. 2013. User Experience + Quality Assurance = Usability. (July 2013). Retrieved February 2, 2016 from http://www.uxmatters.com/mt/archives/2013/07/user-experience-quality-assurance-usability.php

5. Sauli Laitinen. 2005. Better Games Through Usability Evaluation and Testing. (June 2005). Retrieved February 2, 2016 from http://www.gamasutra.com/view/feature/130745/better_games_through_usability_.php

6. Tyler Strickland. 2016. User testing survey results.