ClimbOn – An infinite game with difficulty adjustment

Ricardo Ameixa
INSTICC
Setúbal
rameixa@sapo.pt

Fausto Mourato
DSI / EST / IPS
Setúbal
fausto.mourato@estsetubal.ips.pt

João Morais
ComOn
Setúbal
joao.morais@comon.pt

Abstract
This document describes an infinite game called ClimbOn. The game concept is fairly simple and aims to be suitable for casual gamers that play for short periods, wanting an instant action system with direct and short challenges. The developed system does an automated procedural generation of content with difficult adjustments in order to keep levels challenging to the user and without compromising the final rank system.

Keywords
Casual gamming, Content generation, Human factors

1. INTRODUCTION

As games become more casual and mass oriented, the need to produce new entertaining content quickly enough to satisfy gamers becomes a demanding task for game developers and designers. Procedurally generated levels and difficulty adjustment techniques help companies to achieve these goals, lowering design efforts.

The purpose of this project was the development of a videogame with the application of those techniques in addition to a simple design and game play. The game, called *ClimbOn*, gives to the player the control of an avatar climbing up a building with infinite floors while avoiding falling objects, open windows and other obstacles. The use of procedurally generated content creates a different experience each time the user starts a game. Furthermore, as the user is playing, the game constantly adjusts difficulty to keep the player *on the edge* but not frustrated.

2. MOTIVATION

The term casual gaming refers to games that require a low commitment and skill set by the user. This type of games is normally simple in rules and has a low cost of production and distribution. Most of them are playable for free through web browsers and, nowadays, they are also commonly available on mobile phones. Due to the nature of their quick action game play and their vast yet low skilled audience, normally these games don't use difficulty selection mechanics. Such mechanics are commonly associated with more complex ruling sets and higher commitment games. However, a game with a fixed low difficulty setting soon is forgotten by the player, since his/her skills improve as he/she plays it. The addition of dynamic difficulty adjustment (DDA) systems is a good solution to this problem as it allows the game to evolve with the player to a certain degree and, as an added benefit, it also allows more experienced players to enjoy it without feeling excessive easiness.

3. RELATED WORK

3.1 Procedural generation of gaming content

A popular example of the use of procedural generation and difficulty adjustment techniques is the videogame *Canabalt* [SemiSecret08], developed by Semi Secret Software. This 2D platform game creates segments based on basic buildings and adjusts distances and type of jumps required by the player to how well he/she is performing. The final result is a fluid and immersive experience. A more academic reference worth mention is the work of Compton and Mateas [Compton06], where basic units for 2D platform levels construction are defined as cells. The composition of these cells into patterns and choice of the relations between them define the difficulty of the challenges through the game. By using a physics model and calculating the paths the player can take and based on game play mechanics, the system is able to restrict the levels generated to adequate solutions. The principles presented in this work lead to an implementation presented by Smith et Al. [Smith09].

3.2 Difficulty adjustment

Nicollet et al. [Nicollet04] presented an extensive definition of difficulty concepts in dexterity-based games. Among those principles, one can extract that surprise isn't difficulty. This is as a very valued rule set in guiding what variables should and can be adjusted by DDA systems in dexterity-based games. Also, the work of Hunicke and Chapman [Hunicke04] focus the study of DDA and how these systems create immersive experiences. They make use of a probabilistic technique that measures players' performance based on the difficulty he experiences when given an obstacle. Considering Csikszentmihalyi's definition of flow [Csikszentmihalyi90], they start by assessing what states the player should be kept into and they determine the changes that are needed to force a state change without interfering with the player experience.

4. TECHNICAL ASPECTS

4.1 Game

As previously stated, we developed a game with one single level with one building constituted by infinite floors. The player has to climb it up while avoiding falling objects or other obstacles. Each floor has been represented as a set of three windows, thus this is a configurable parameter. The game automatically increases the player speed upwards, leaving to the user the control of the side to side movement as a way to avoid obstacles, in order to continue playing. The basic score of the player is determined by the distance climbed: the bigger the distance, the bigger the score.

4.2 Level Generation

To guarantee game play continuity, the game has to assure that the player has always a valid path to follow. In order to accomplish this goal, the game takes into account that the player is not able to transverse a two window space in just one floor, thus it needs to have its path position either to be placed in the same position or on an adjacent position relatively to the last floor created. In the calculation of the path, we used a scaling progressive percentage factor that can be influenced by the DDA system. A new set of floors is created each time the player enters the current set. Having this buffer gives the game time to analyze the player performance and generate the transition set seamlessly. The obstacles placed outside the path are randomly chosen to create diversity in the facade of the building. Some of the obstacles may even be created as path looking in order to create visual noise as an added difficulty element.

4.3 Difficulty adjustments

To be able to adjust difficulty, the game needs a way to measure the player performance. This measurement is done by directly relating closer positioning to the obstacles with the increased difficulty the player has to react to said obstacles. In other words, the game measures the average distance between the player and the obstacles in his neighborhood at all times and in case the player is too close for too long, the DDA system will change the flow of the game. For that, it just needs to change the progressive percentage factor in order to adapt the place where the next path window is more willing to be. As a simple heuristic, keeping the avatar in the same row is easier than moving to an adjacent one. Later adjustments are also done, for instance, adding more or less power ups, changing their power type or increasing the player upwards speed. Having a bigger space between movements on the path or a slower upwards speed, gives the player more time to react.

5. RESULTS

In this section we present the result of our game. In figure 1 we can see the main look of the game, which is by this time being tweaked in its design, especially in which concerns to models and textures. It is possible to observe three effective floors generated with low difficulty. With this low difficulty level, only a small amount of obstacles (in this case: windows) were generated.

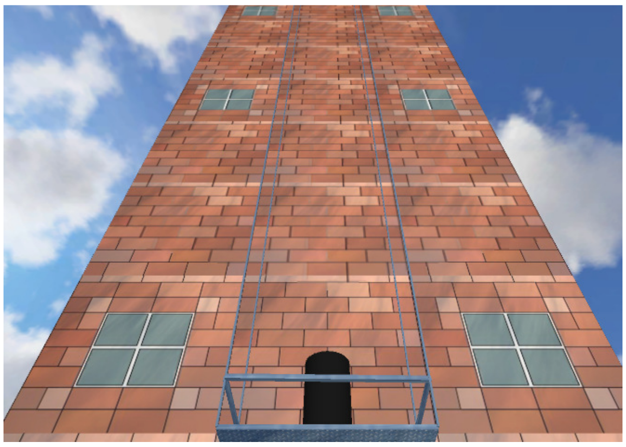

Figure 1 – Screenshot from our prototype: *ClimbOn*

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

We presented a game called *ClimbOn* that generates infinite self adjusted levels to provide challenges to casual gamers. This work represents a practical implementation of two main concepts: procedural generation of game spaces and difficulty adjustment. Our primal tests with a restricted set of users showed that the difficulty adjustments are transparent to the user, which realizes that the game gets harder with time but without strong transitions.

6.2 Future Work

The main purpose of this article was to present our approach with a concretization in a real game, having the main focus in the procedural generation and the product result. However, we believe that is important to expand some of the principles, in particular to effectively use a metric for difficulty in this game and extract data, such as losing probabilities, among others, from game play in order to extend and validate some of our assumptions.

7. REFERENCES

[Csikszentmihalyi90] Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. NY: Harper Collins, 1991.

[Compton06] Compton, K., Mateas, M.: Procedural level design for platform games. Proceedings of the 2nd Artificial Intelligence and Interactive Digital Entertainment Conference, 2006.

[Hunicke04] Hunicke, R., Chapman, V.: AI for dynamic difficulty adjustment in games. Challenges in game artificial intelligence. Pittsburgh: AAAI Press, 2004.

[Nicollet04] Nicollet, V. Difficulty in Dexterity-Based Platform Games. Gamedev.net. (May 22, 2010). <http://www.gamedev.net/reference/design/features/platformdiff/>

[SemiSecret08] CANABALT: (Browser Game) Semi Secret Software, 2010. Last Accessed: May 22, 2010 <http://canabalt.com>

[Smith09] Smith, G., Treanor, M., Whitehead, J., Mateas, M. 2009. Rhythm-based level generation for 2D platformers. *Proceedings of the 4th international Conference on Foundations of Digital Games* (Orlando, Florida, April 26 -30, 2009). ACM, NY, 175-182.