Entertainment Computing 16 (2016) 81–93

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

TIṬAL – Asynchronous multiplayer shooter with procedurally generated maps ☆,☆☆

Bhojan Anand *, Wong Hong Wei

School of Computing, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 1 March 2015
Revised 21 January 2016
Accepted 19 February 2016
Available online 7 March 2016

Keywords:
Game
TIṬAL
Prodecural content generation
Game map generation
Bots
Artificial intelligence

ABSTRACT

Would it be possible to bring the promise of unlimited re-playability typically reserved for Roguelike games to competitive multiplayer shooters? This paper tries to address this issue by proposing a novel method to dynamically generate maps at run-time almost as soon as players press the Play button, while ensuring the features what players would expect from the genre. The procedures are simple and practically feasible to be employed in actual computer games. In addition, the work experiments the possibility of incorporating asynchronous game-play element into a multiplayer shooter with human imitating bots where the players can let their bot/avatar replace them when they are not around. The algorithms are implemented and evaluated with a playable game. The evaluations prove that playable 3D dynamic maps can be generated in order of seconds using game context data to initialise the parameters of the algorithm. The paper also shows that asynchronous game-play element is a possible feature for consideration in next generation multiplayer shooters.

1. Introduction

This paper demonstrates a novel method for procedurally generating dynamic maps at run-time at the click of the 'Play' button using game context data and experiments the feasibility of bringing asynchronous game-play element in next generation multiplayer shooters. We have created an experimental game entitled *TIṬAL*, a Multiplayer Shooter in the form of a First-Person Shooter (FPS) with novel approaches in the design of procedural maps and human-imitating bots.

We limit our scope to the game mode **Capture and Hold**, popularised in games like Killzone [1]. Every player belongs to one of 2 teams. Littered around the map are *flags* that are captured by placing players in close proximity. A captured flag (can be recaptured) lowers the *team points* for the opposing team continuously. A team point of 0 signals the victory of the opposing team. To win, teams have to actively defend and pursue flags.

1.1. Procedural content generation

Procedural Content Generation (PCG) is the use of algorithmic means to create content [2,3] dynamically during run-time. Instead of trekking the same grounds which gets stale with time, PCG promises a more novel experience every playthrough. PCG was utilised in games as early as 1978. A notable example is *Rogue* [4,5] which spawns a new genre known as Roguelikes. Core features include randomly generated levels, item locations and so on. PCG's influence extends to racing games like *Gran Turismo 5*, which procedurally generates its tracks [6]. First Person Shooter (FPS) *Borderlands* series procedurally generates weapons [7].

1.2. Multiplayer shooter

We believe that an exception to PCG lies with environments/maps for multiplayer shooters such as Battlefield [8], which remains one of the most popular genres to date [9].

The reader should note the difference between a multiplayer and a standard shooter, which often follows an approximately linear path. The competitive nature of multiplayer shooters brings additional challenge of ensuring fairness through the positioning of strategic points. Moreover, players typically have more freedom to move around the map. It is for these reasons the designers often take a different approach to craft multiplayer maps.

☆ This paper has been recommended for acceptance by Nikitas Marinos Sgouros.
☆☆ This paper is part of the virtual special issue on "Selected Papers from the 2014 International Conference on Entertainment Computing", edited by Dr. Nikitas Marinos Sgouros and Dr. Matthias Rauterberg.
* Corresponding author.
E-mail addresses: banand@comp.nus.edu.sg (A. Bhojan), wei@nus.edu.sg (H.W. Wong).
URL: http://www.comp.nus.edu.sg/~bhojan (A. Bhojan).

http://dx.doi.org/10.1016/j.entcom.2016.02.002
1875-9521/© 2016 Elsevier B.V. All rights reserved.

In this paper, we present a novel method that uses carefully selected game context data to generate dynamic maps almost as soon as the player press the ‘Play’ button, without compromising the expected features of a good multiplayer shooter environment. The method is simple, practically feasible and we have implemented and evaluated it with the test game. We call our method as ‘game context aware dynamic map generation’ method.

1.3. Human imitating bot

Recently, we are exposed to the concept of *asynchronous multiplayer*. Unlike games which demands players to play simultaneously, asynchronous multiplayer games can be played at any time. In Draw Something [10], the other player does not need to respond instantly in every round.

TIȚAL experiments with the possibility of incorporating such an element into a multiplayer shooter. To allow this, the player can play against a bot that imitates the behaviour of a teammate, similar to Forza Motorstorm [11].

We also present a method for creating bots that take after the behaviour of a player. In contrast to previous methods that records huge data containing movement and action patterns to mimic the behaviour, we learn and store only the key parameters that are meaningful for the given game genre to mimic the behaviours in key decision points of the game. By using such game context aware learning, the storage and computation requirements are significantly reduced while producing human like behaviour. Like in PCG, the method is practically feasible and has been implemented in a multiplayer shooter and evaluated with a group of users.

1.4. Overview

In the rest of this paper, we first discuss the motivations for this work in Section 2 and then related works in Section 3. In Section 4 we state the design goals of our map generation method before describing how it is achieved in Section 5. We then explain the design of the human imitating bots in Section 6. Implementation, results and evaluations are described in Sections 7–9 respectively. Finally, we discuss the limitations and future works in Section 10 before concluding with Section 11.

2. Motivation

In this work, we are attempting to automatically create playable, balanced (fair) and interesting maps for multiplayer shooters, with a novel approach built using Search-based PCG [2]. While PCG is used by some multiplayer shooter game designers, who would procedurally generate maps and then manually tweak them to ship with the final product, our goal is to remove the human intervention completely. This means that the generation should be completed within a span of seconds, or else the patience of the player could wear thin. If we are successful, the development time and cost needed to create maps for similar games could be drastically reduced. The result would be increased longevity that stems from the near limitless amount of maps for players to play in.

We also aim to create Human Imitating Bots to incorporate asynchronous gameplay to a new genre. Unlike most works which aims to create human-like behaviours using a huge repository of player’s play throughs, we hope to imitate the behaviour of a single player and apply it to any map. Its success could bring about something fresh, allowing players to create a ‘doppelganger’ of themselves and watching it climb the ranks as it compete with others. It also reduces the time commitment players required for such games.

3. Related work

3.1. Procedural generation of map

Güttler and Johansson [11] identified basic spatial properties of multiplayer FPSes and proposes heuristics for better level design. In addition, the insights provided by industry leaders on design of multiplayer games [12–14] assist the formulation of our design goals.

Search-based PCG (SBPCG), an approach to PCG, was introduced by Togelius et al. [2]. We will be employing a similar approach in our solution.

Togelius et al. also manages to procedurally generate tracks for a racer [15] and maps for Starcraft [16]. In both cases, SBPCG is used with a simulation-based fitness function. Kerssemakers et al. [17] introduces a procedural PCG generator to generalise PCG with the use of a simulation-based fitness function. However, the use of simulation-based fitness function is not suitable for our goals, due to the time it takes to create a map is long and it is not suitable for practical implementations due to the strict real time requirements imposed by the games and game players.

Yeh et al. [18] presents the Markov chain Monte Carlo (MCMC) algorithm to create open world areas which are relatively easy to create when compared to complex closed area maps which basically represent big buildings. Michael et al. [19] proposes a method of tiling entitled Wang Tiles for image and texture generation. Merrell et al. [20] identifies furniture design guidelines for indoor areas and proposes ways to suggest layouts that fit those guidelines. Similarly, a method for constructing concrete-based buildings is introduced by Liu et al. [21]. However, their focus is only on generating components for the game map, but not the entire map.

Work done by Cardamone et al. [22] to evolve interesting maps for a FPS leveraging on SBPCG is a great starting point. However, work needs to be done to ensure that the map can be generated within seconds. Moreover, the maps that are generated are seemingly low on navigability and aesthetics, which are basic features of any good multiplayer game. In contrast, navigability and aesthetics are part of our design goals.

3.2. Human imitating bot

The work by Ortega et al. [23] introduces us to imitating human play styles in Super Mario Bros.

Thurau et al. [24] created an AI Bot that captures the movement of a human player in a FPS. It relies heavily on how players move and then replicating it in the same map. For our game, we also aims to create maps dynamically Therefore, we want to bring this behaviour to any maps even if part of the accuracy has to be sacrificed. Moreover, it relies on massive existing player data to craft them, which differs from our goal of obtaining a good model of a single player.

Verweij [25] shows us techniques in implementing multiplayer bots which drives the design of our AI. The paper by Waveren [26] details the Quake III Arena Bot which is mostly state-based and therefore may lack the nuance needed for mimicking human behaviour.

Schrum et al. [27,28] introduces us to the UT2 bot for Unreal Tournament 2004, which maintains life-likeness after evolving with a set of human play traces but is not exclusive to a single human player.

4. Map design goals

We describe what we want to achieve by identifying elements of interesting maps (of multiplayer shooters) so that they can be incorporated into our design.