Balanced First Person Shooter Level-Generator

James Moran

# Overview

Having played various First-Person Shooters (FPS), since my early teenage years, as well as having seen the competitive play of more recent FPS titles (such as Counter Strike: Global Offensive (CS:GO)), it has become apparent to me, that there are certain FPS tiles, which have levels, that have not completely (or on rare occasions, at all), considered balanced level design. That is in a nutshell, levels that would not favour one side or the other, on an overall basis (with advantages and disadvantages for each team, given where they start in a level, that the Players must overcome).

For context on this issue, there are various groups of Players who feel as though certain levels of certain FPS titles, favour one side far more greatly than the other. This leads to Players having feelings of unfairness, in relation to how they are at a disadvantage from the start of a game (irrespective of Player skill-level). This is in respects to the paths Players can take through the level, along with where they would have to look to find enemies at certain points in a path, or from entrances/exits to/from a path, as well as entrances to other paths. (LevelCapGaming, 2014)

The scope of the project, will be broken down into three main phases (with two bonus phases):

* Greybox Phase: In this phase, the initial framework for the level generator will be implemented, so that it can produce the geometry for the level (such as the walls, entry and exit points, as well as obstacles to provide cover)
* Collision Bounds Phase: After the geometry for the level has been generated, one could import this level, as a mesh into a game project. The problem with that is, the project handling system (e.g. a game-engine), would consider the level asset as one asset, applying a collision box or sphere, that envelops the whole level. For this phase then, the Level-Generator would have to create collision bounds for each piece of geometry it has generated in the first phase (for walls, obstacles, doorways and other entry/exit points to name a few)
* Texture Phase: After a level with suitable geometry and collision bounds has been generated, comes that of applying appropriate textures to the geometry, as per the setting of the game’s level (such as clinical, office related textures, for the corridors/cubicles and walls, of an office building)
* Bonus Phase One: Prop Phase: This is a bonus phase (a stretch goal), as it is not critical to the purpose of the project and will only receive implementation, if there is suitable time for such, after completing the first three phases of the project. In this phase, props (either dynamic or static) will be added to the level accordingly (such as chairs, desks, stationary equipment, water-coolers, given an office setting). These can either be destroyed, moved or broken through, to remove them as additional obstacles in the level.
* Bonus Phase Two: Lighting Phase: This is a bonus phase (a stretch goal), as it is not critical to the purpose of the project and will only receive implementation, if there is suitable time for such, after completing the first three phases of the project. If this phase is not undertaken in the project’s development timeline, an ambient (global) light source, will be applied evenly to the whole level, but if this phase is undertaken, then appropriate lighting for each section of the level, will be generated by the Level-Generator (coming from ceiling lights, lamps, torches, external sources (such as from windows), as well as from certain other props in the level (such as a mobile-light source, provided to one or both of the teams)

# Overall Aim(s)

Considering this, the overall aim of the project will be that of generating a level, for an FPS, that has an interior basis (such as a cave or an office building), with one main degree of level gradient (as this level will take place on one level of a building, with minor ascent possible via obstacles that the Player can get on top of, if accommodated for in the level). In addition, the generator will produce a ‘balanced’ level, that takes account of various properties for a balanced level, such as movement paths through the level (to objectives or otherwise), choke-points in the level, as well as where Players will want to check for enemies, given the route they have taken through the level.

# Initial Objectives

When thinking of some initial objectives for the project, the following come to mind:

* Consider relevant literature, for algorithms that one could utilise for (procedural) level-generation (literature considered for review and initial lines of research, will be listed in the next section)
* Using the chosen level-generation methodology, implement this method to the extent that it will generate the level’s ‘floor’ (to be used as a basis for all other parts of the level)
* Following on from the previous point, implement functionality to allow the system to generate bounds, surrounding the base floor of the level (such as a wall on each of the four sides, to represent a level of a building)
* After this, then move onto to implementing functionality to allow the system to generate the main geometrical features of the level, contained within the bounds of the level (such as rooms with their subdivisions for a building)

# Relevant Literature

I intend to use the following sources to guide me in the implementation of the project (This is subject to change):

* Procedural Content Generation in Games (Computational Synthesis and Creative Systems) – Noor Shaker, Julian Togelius and Mark J Nelson
* Level design: Processes and experiences – Christopher W. Totten
* The Science of Level Design: Design Patterns and Analysis of Player Behavior in First-person Shooter levels – Kenneth Hullett

# Project Management Approach

The project will receive management by following a notable portion of sensible project management techniques. This process is initiated by identifying certain characteristics of the project, these are:

* The project’s objective
* When the project is to have been completed by (scheduling)
* Project Complexity
* Tasks of the project, the time required to complete these tasks and how one should complete a project task
* Available Resources
* Organisational Structure
* Information and Control Systems

(Sean Maserang, 2002).

The techniques used to manage the project, are most notably that of the Program Evaluation Review Technique (PERT) Charts and Gantt Charts, for suitable time management of the project, to keep on track with the project, not finishing tasks too late or too early (Sean Masterang, 2002).

The Software Development Methodology that will receive utilisation in the project, is that of the Spiral Methodology. This was chosen as it is an iterative development cycle; getting a concrete baseline of the system’s requirements, before creating the initial and subsequent prototypes of the system, of which, the final prototype becomes the final version, which is then thoroughly evaluated and tested (IT Knowledge portal, 2017).

A high-level overview is shown below (IT Knowledge portal, 2017):

The project management tools that I will use, are that of Microsoft Project, to construct PERT and Gantt Charts, along with Microsoft Visio, for creating Flow Diagrams/UML Activity Diagrams, UML Class Diagrams etc. Such as the diagram shown to the left.

A Trello board, will also receive construction for the project, in order to make sure that certain important tasks, receive the necessary priority required to conduct the necessary sub-tasks for the project.

Along with using a Trello board, source control for the project’s source files will receive utilisation as well, specifically, the GIT source control system. This will receive usage in of itself, with an appropriate client. The client chosen for the use of the GIT source control system is that of GitHub. (© 2017 GitHub, inc.)

# References

1. LevelCapGaming, 2014. *FPS Level Design – LevelCap Bashes BF4 Map Design* [viewed 02/12/2017]. Available from: <https://www.youtube.com/watch?v=FN7iLKUR8eY>
2. Sean Maserang, 2002. Project Management: Tools & Techniques [viewed 26/04/2017]. Available from: <http://www.umsl.edu/~sauterv/analysis/488_f02_papers/ProjMgmt.html>
3. IT Knowledge portal, 2017. Software Development Methodologies [viewed 27/04/2017]. Available from: <http://www.itinfo.am/eng/software-development-methodologies/>
4. GitHub, 2017. *Built for developers* [viewed 05/12/2017]. Available from: <https://github.com/>