**1. Introduction**

* 1. **Motivation**

From the very moment I discovered videogames, I have been inevitably attracted to their maps and environments.

Those maps, as I would discover on my own while creating levels, are crafted by employing vast amounts of design guidelines and all sorts of tricks.

As a map creator, I thus concluded that, in order to speed up the process and better the result, I could apply those techniques by employing an automated workflow.

Said workflow would permit level designers analyze their work by using AI capable of translating those design guidelines into map navigation and feedback.

Since *First Person Shooter* videogames constitute a genre that I am familiarized with and, on top of that, one that is straightforward in terms of testing, I decided to delimit my thesis inside this context.



*Top-down view of an fps level of my own… But, how do I make sure it works?*

* 1. **Problem Formulation**

Traditionally, so as to test and iterate their work, level designers had one tool at their disposal: human playtesting.

This concept, however, became quickly tied with a set of hindrances:

* Humans give highly subjective and likely contradictory feedback.
* Human playtesitng is slow in comparison to computation.
* Humans require accomodation.

Therefore, videogame companies have shyly started to combine human playtesting with AI automation.

Casual and simple game sagas such as *Candy Crush* employ AI that can solve entire levels by using algorithms of choice that evoke different player playstyles.

Other videogames and software applications, instead, introduce *machine learning* to continuously improve their agents towards an optimal behaviour.

Nevertheless, in the current 3D FPS map-making scene, there is an unrgent need of a generic tool capable of analyzing every common level design element involved and shared within the genre so that a sensible human-like feedback can be obtained.



*Candy Crush uses Player Personas*

* 1. **General Objectives**

The main goal that I strive towards is to develop a **theoretical model** around **FPS level design parameters** (monster placing, difficulty curve, heatmaps…) and how to **analyze them automatically**.

Only then, the second part would be to develop a **prototype** where FPS level designers can input their map and **receive human-like feedback** for further level iteration.

With that purpose in mind, two requisites need to be fulfilled:

* Publish a generic, easy to setup **Unity tool**, free at the Unity Store, that can function within common FPS settings.
* Run a vast amount of **simultaneous tests**, each of them mildly altered, and draw conclusions from potentially polarized data.
* Develop an **AI capable of navigation**
* Incorporate a **basic FPS enemy** type and **simulate combat instantly**.
  1. **Specific Objectives**

In order to achieve the above objectives, these functionality goals are to be considered:

**Principal**

* Deploy a **real-time output window** that shows information about the simulations.
* Employ a Unity **Test Runner** package to tag code fragments to be tested and to then handle the parallel simulations.
* Approach navigation and level evaluation from a **3D tile grid** perspective.
* Have an algorithm that attributes **each tile a value**, based on multiple design variables, so that AI agents can effectively choose where to move.

**Secondary**

* Generate **AI capable of common FPS actions**, including: moving, shooting, reloading, picking items… That is also **aware of their surroundings** (sound cues…) **and status** (HP…) so that **different tactics** are employed.
* Code the most **recurrent FPS** enemies, weapons, items…
* Offer an array of **customizable parameters** in regards to the above elements and player AI.

**Optional**

* Center part of the player AI’s code around **human cognitive theory** so that decisions made while navigating the level are legitimate.
* **Evolve each AI player**: the agent knows more about the level each time a simulation is completed (*machine learning*). **Embed** some agents within the package and **store** them on demand.

* 1. **Project Scope**

The theoretical part of the thesis can relate to **all traditional FPS game levels**, ranging from relics like *Half-Life* to a more modern *CSGO*.

In said section, one of the key aspects to analyze is the **level design commonalities** between them, which will be highlighted and will translate into the aforementioned model.

The ideal outcome of this research is both public documentation about **translating** **design variables** **into automated playstesting** and also the **chosen approach for each variable** in the case of my prototype.

Once the prototype is finished, the bulk of **indie developers** that utilize Unity will have a generic, free tool, that, with an intuitive setup, is able to automatically test their FPS levels.

**2. State of the Art**

**3. Project Management**

**4. Methodology**

**5. Development**

**6. Bibliography**

Candy Crush AI: <https://deepai.org/publication/automated-playtesting-of-matching-tile-games>