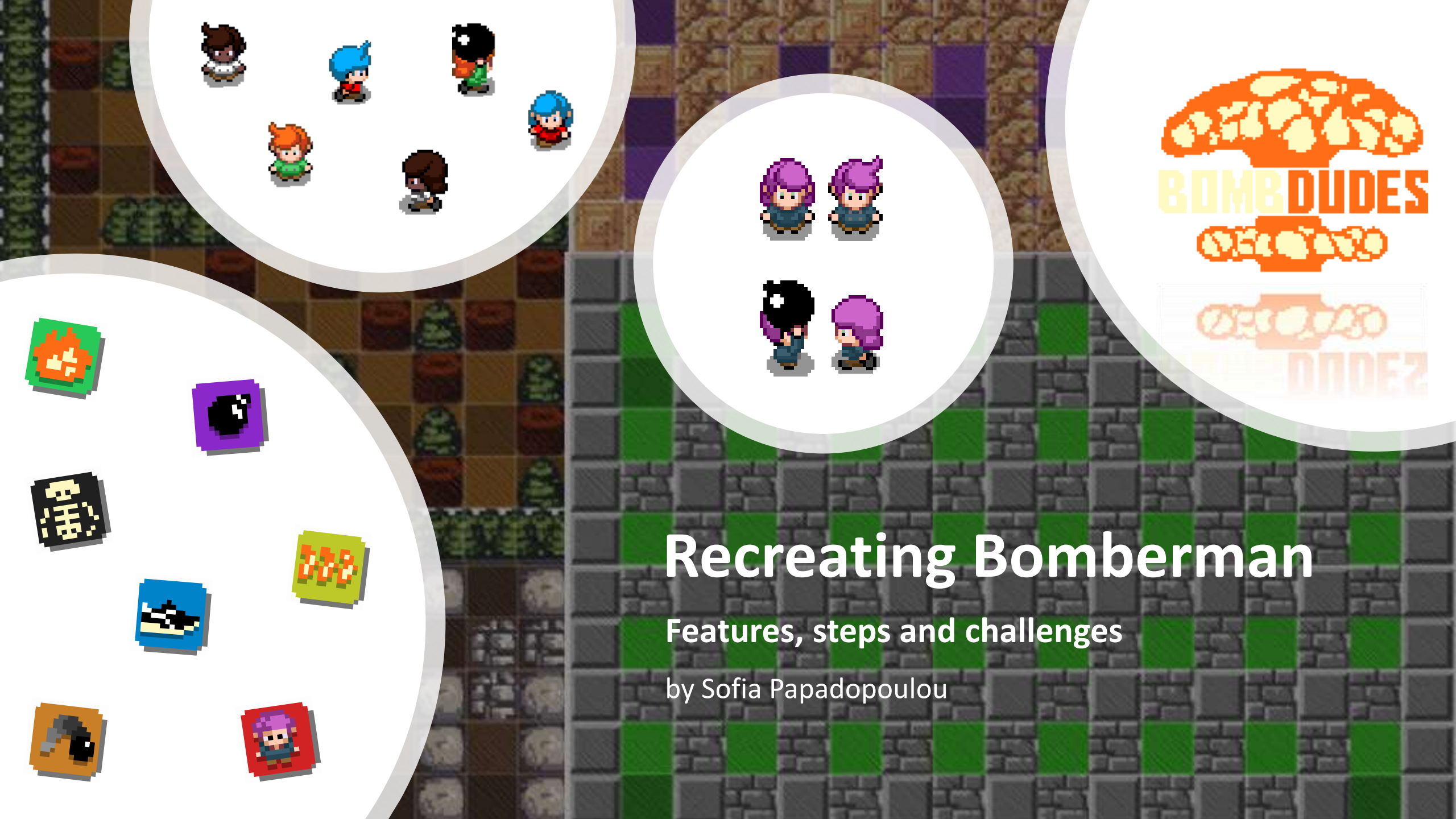


Recreating Bomberman

Features, steps and challenges

by Sofia Papadopoulou

Background

Many games are implemented daily, but to my knowledge, there is no literature presenting and justifying the steps a game developer follows to complete a game.

There are books, like ‘Level Up! The Guide to Great Video Game Design’ by Scott Rogers, that analyse best practices on every game feature, but it is quite different when a developer is called to build a game.

- ✓ Break a game into tasks
- ✓ Organise them or divide them into sub-tasks
- ✓ Make decisions on the best way to implement something

Bomberman is one of the longest-running series, offering a great number of gameplay features and alterations. However, existing literature work focuses almost exclusively on Artificial Intelligence techniques, such as the da Cruz Lopes and Manuel António’s ‘Bomberman as an artificial intelligence platform’. In this project, Bomberman was selected due to its popularity and the lack of previous research in game developers’ approach to the development process.

Introduction

What is the ‘Bomberman’ game?

- First release in 1983, by Hudson Soft
- Also known as Dynablaster and Atomic Punk
- Long-running series, on many different platforms
- Gameplay:
  - Strategy, maze-looking game
  - Usually has 2 modes: Adventure and Battle
  - Main goal is
    - either to kill all the enemies and find the exit (adventure), or
    - kill all the other players and be the last one standing (battle)
  - Weapons to use: bombs
  - Many power-ups such as: bigger explosions, more bombs available, speed up etc.
  - Levels: maze-looking maps, with static (walls) and destructible objects (bricks, barrels, etc.)
  - Threats: enemies’ touch (adventure) and bombs explosions (player can die even from its own bombs)

Implementation and Challenges

Let's consider a scenario where the description on the right is given to a developer as specifications for a game.
The information that can be extracted is the following.

Gameplay Features Needed:

1. 2D graphics and camera handling to create a 2.5D projection for single-player mode
2. artificial intelligence (AI) for bots (human playing behaviour) and monsters (similar behaviour to Bomberman's enemies)
3. animation on 2D sprites for player, bot and monster movement, or bomb's detonation (similar to Bomberman game)
4. user interface (UI) for game mode selection, character selection, bots addition and players' lives
5. audio for making the game more vivid and engaging
6. tilemaps and grid for levels' creation

"A 2D Bomberman-type game, with both single and multi-player mode. The maximum players will be 4, but if less, players will be able to fill in the empty positions with bots. The camera will be tilted and following the player on single-player mode, but it will be top-view on multi-player. In single-player mode, the player will select between the Adventure and the Party mode. Each level map will be a 17 by 13 grid, made with blocks of 16 by 16 pixels."

Implementation and Challenges

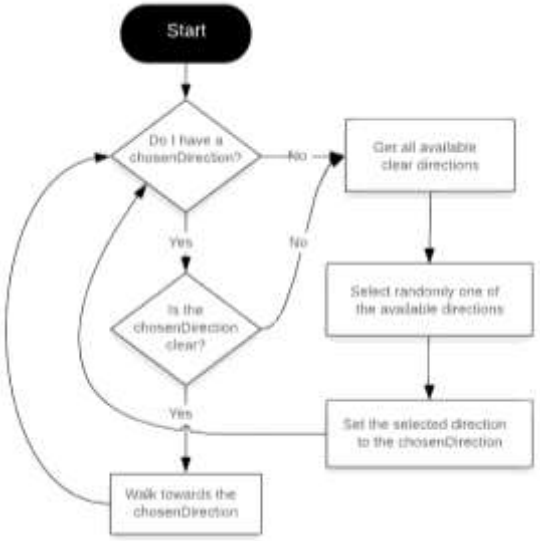
Breaking the game into tasks is a challenging thing, and thus we present and indicative list of steps for a game developer to follow:

- ✓ Create (at least) one level with Tilemaps and a Grid, to have an environment to work on, build the ground, destructible and indestructible objects on separate tilemaps, for easy processing
- ✓ Create the objects needed, along with their functionality (Player, Bomb, Power-ups, Bot, Monster)
- ✓ Create generators for dynamic objects that are altered per level: Power-ups spawner, Levels spawner, Monsters spawner
- ✓ Create the UI: Start screen, Game Over screen, Mode Selection screen, Next Level Screen, Winner Screen, Character Selection screen, Game Completed screen
- ✓ Create a camera manager
  - ✓ for Adventure mode: 3D perspective, attached and centred to player object, an array of selected objects that will be rotated on a -30° angle to face the camera (player is included in the array)
  - ✓ for Party mode and Multi-player: 2D orthographic, attached to root, top-view camera centred on the whole map
- ✓ Add sound effects and music, synchronised to actions happening in the game
- ✓ Extras for making the game more interesting: screen-shake on explosions, arena shrinking gradually when multi-player or party mode.

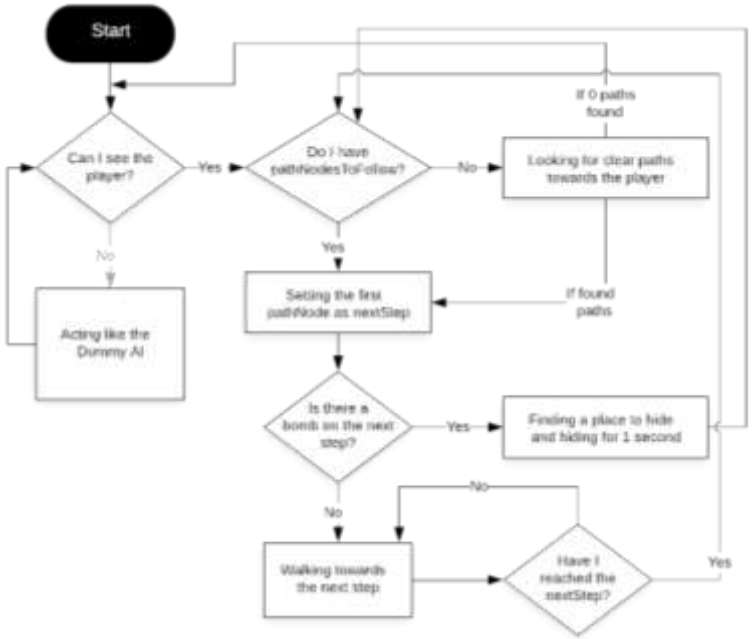
Implementation and Challenges

During the implementation, some challenges occurred, as it was expected. Bomberman is a game that requires some well-implemented AI, because of the need for continuous path-finding. A developer is called to achieve good performance by handling properly and limiting the path-finding as much as possible. In this project, the A* algorithm was selected as the most appropriate one for path-finding. The following three types of AI were used:

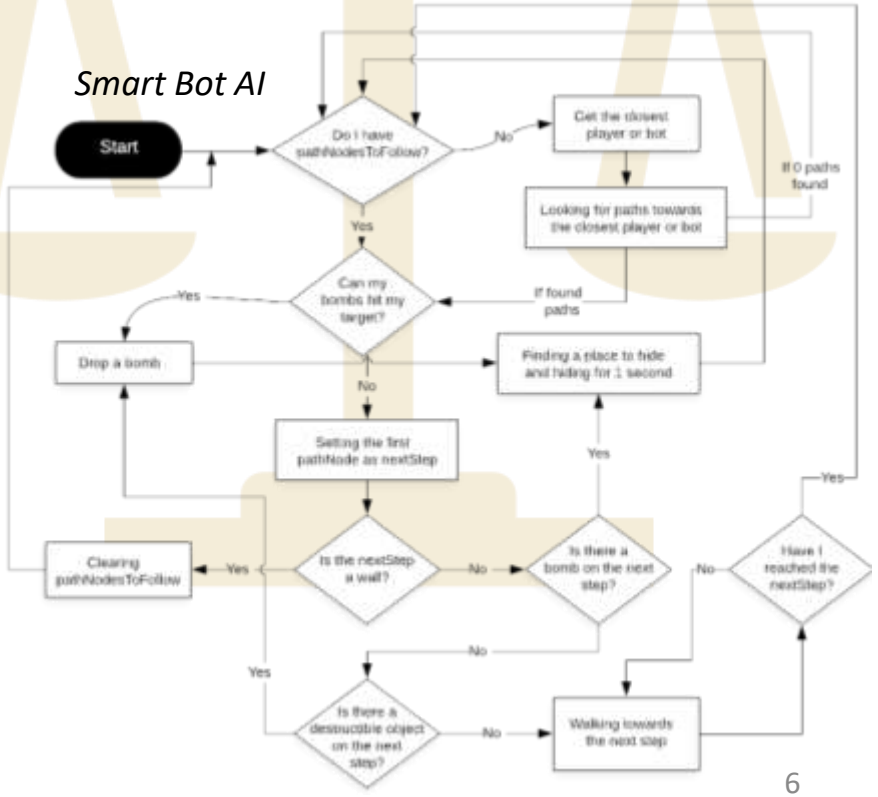
Dummy Monster AI



Smart Monster AI



Smart Bot AI



Evaluation and Analysis

All successful games share some common characteristics:

- good performance on every platform they are created for,
- easy interaction with the user
- and satisfying audiovisual fidelity.

Common issues that cause unpleasantness to users are:

- Slow framerate
- Memory consumption
- Game pausing

Game programmers are responsible for delivering well-written code that prevents memory leaks, CPU overheating, or unhandled code exceptions, that may result in a game's failure.

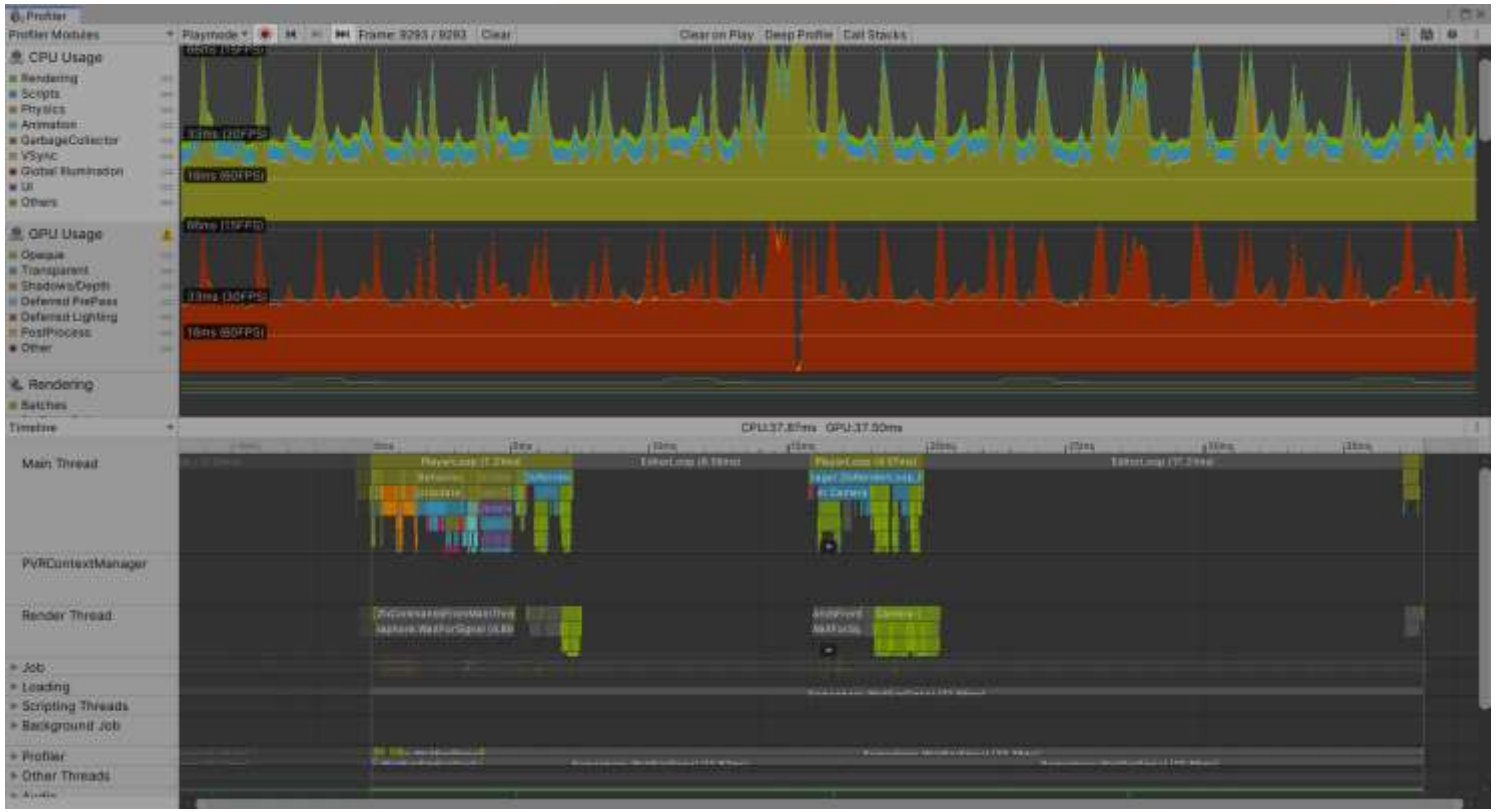
Realism in games is a very important aspect and it is not related just to the graphics. Physics, audio and user interface, affect the whole game experience as well. If a game 'feels' nice, the developer is on a good path, but this is not enough. Performance should be evaluated and improved if needed.

Evaluation and Analysis

Most game engines provide profiling tools that are extremely helpful and every game developer should be familiar with.

The following screenshot comes from Unity's Profiler window and was taken in a laptop with 64-bit Windows 10, a 4-core CPU at 1.80GHz and 8GB RAM and 2 GPUs with 8GB total GPU memory.

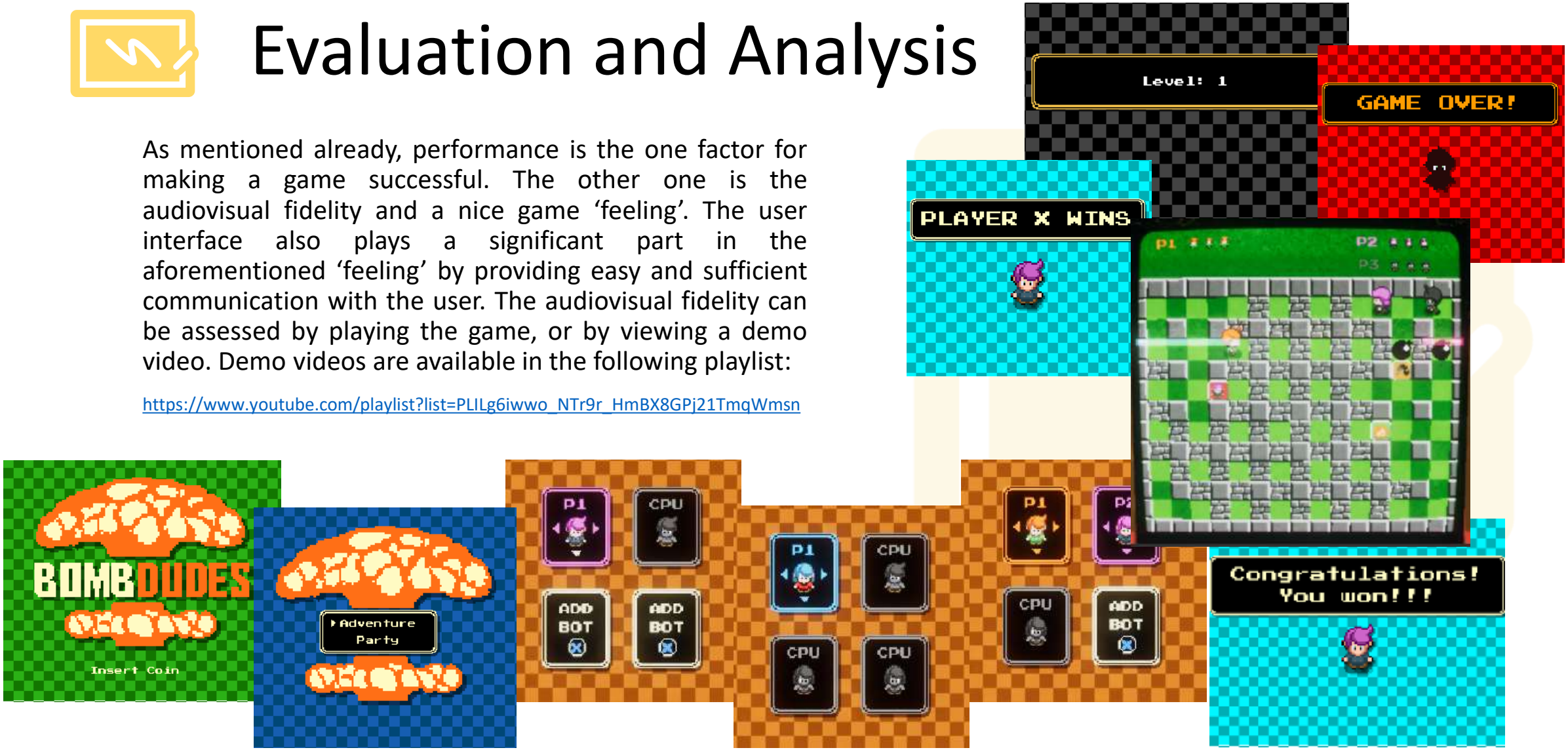
However this image cannot be indicative exclusively for the 'BombDudes' game (the Bomberman-type project), because it was created as a part of a larger industrial project, that is also loaded on the background, while this game is running.



Evaluation and Analysis

As mentioned already, performance is the one factor for making a game successful. The other one is the audiovisual fidelity and a nice game ‘feeling’. The user interface also plays a significant part in the aforementioned ‘feeling’ by providing easy and sufficient communication with the user. The audiovisual fidelity can be assessed by playing the game, or by viewing a demo video. Demo videos are available in the following playlist:

https://www.youtube.com/playlist?list=PLILg6iwwo_NTr9r_HmBX8GPj21TmqWmsn



Conclusions

- Building a game from zero, especially for an inexperienced developer is not a ‘piece of cake’. Re-creating an existing game as a modern altered version is slightly easier because the developer has a better-formed image of what the game will look like.
- Digging in given specifications or an idea that someone has in order to extract the gameplay features is a challenging task and comes easier after practice. Requirements usually contain keywords, that developers learn to translate in specific features.
- An intriguing part is the process of breaking the whole project into tasks, assessing priorities and organising the development time.

The current project provides an indicative method of creating a game, based on suggested best practices found in previous literature work, in combination with personal experience gained while working in the gaming industry.

PRACTICE MAKES PERFECT