Game Development - Assignment 2

Overview

Building on top of our previous assignment, we are going to add enemies and limit the frames per second, while we understand how we are distributing each millisecond each frame time over the different systems.

Forking from other student's previous code

Students have the option to start from the code from other students just on the moment of delivery on the previous. Be sure to mention this in the README and properly fork the other team's repository exactly at the moment of delivery of previous assignment. Forking late will mean not accepting the delivery.

Content

Expanding the platformer from the previous assignment we need to add:

- Flying enemy type that can pathfind to the player avoiding non-walkable areas.
- Walking enemy that can pathfind to the player. It is not needed that the enemy can jump (although is encouraged) but it should detect that it can reach the player by normal walking and falling down to other platforms.
- Now saving/loading must save each enemy state. Enemies normally have a range of perception and not react to the player until they are close by. This is up to you.
- By default, game should be capped to stable 30 frames per second *without vsync*. The title of the window must show:
    - FPS / average FPS / MS of the last frame (Cap on/off + Vsync on/off)
- Game should have all it's movement normalized using **dt**, so in slow/fast machines it would keep the same movement speed.
- Internally, it should use a structured entity system that should be described in an UML file (pdf) delivered in the same folder of the release. The document should NOT be a copy paste of the code but a description of its most important parts.
- There must be a way to destroy enemies, being jump onto them, shooting / using some weapon ...
- The code should have **Brofiler** integration and allow measurement of its core modules, with a big distinction between time to swap buffers (graphics), pathfinding code and other logic code.

Minimum debug Keys

F1/F2 Start from the first/second level	**F3** Start from the beginning of the current level
F5 Save the current state	**F6** Load the previous state (even across levels)
F9 To view colliders and pathfinding	**F10** God Mode (allows to fly around)
F11 Enable/Disable FPS cap to 30	

Submission rules

The delivery must be a **Win32 Release** zipped in its folder inside *"Second Assignment"* named after your game (e. G. Ultra_Mario_Bros.zip):

1. Delivery should contain only the minimum assets needed to run the game that should be compiled in Release. Only the required dll to execute the game should be there (with exception of the UML scheme delivered as a **pdf** file that describes the entity system used).
2. Your code good .gitignore and well commented, small commits on your changes over time.
3. The repository under github.com must contain a copy of the build under the Release section.
4. There must be a text file called "README.md" containing: info about the game, authors, a **link to the github** repository and a *license*. Any special instructions for execution should be included in this file, as well as any system that you think could add to the innovation grade. Also remember to add a Credits section to mention all source of art / audio you used. Also describe what each team member has been doing for this specific delivery. The commits should show this work.

The assignment must be submitted before **November the 17th 23:58** *(folder closes automatically)*

Grading Criteria

To **accept** a submission for grading, it must comply with:

1. It followed the submission rules stated above.
2. The code compiles and uses only english.
3. It should be **original**. If code is found to be copied across teams, it won't be accepted.
4. The game did not crashed while testing.

Once accepted, the criteria is as follows:

- 50%: C++ code is clean, well structured and easy to read.
- 30%: Enemies can find their way around the level easily and *Brofiler* gives all sensible information about how the frame time is distributed over the code. Sprite movement is normalized and the entity system structure makes sense taking in account the game context.
- 20%: Dodging / Fighting enemies is fun and has a balanced difficulty that provides a good experience for the player.

***Note**: In case of a great imbalance in work between team members, teacher can decide to downgrade an individual score.*

***Note**: Remember that any experimental system could potentially impact the innovation grade. If you think you have some system that falls in this category, clearly mention them in the README.*

Innovation Examples

- Extra game features like teleportation 5-20%
- Skills with cooldown 5-20%
- Checkpoint autosave 30%
- Culling collisions 15%
- Ramps / diagonals / sliding 5-25%
- Spatial Audio 15%

Useful Links

Adapting A* to platformers
Brofiler homepage