Introdução à Matemática e Física Para Videojogos I - Final Project

This is a very rudimentary, wireframe 3d engine.

This serves as a basis for the "Introdução à Matemática e Física Para Videojogos I" course, on the Licenciatura em Videojogos da Universidade Lusófona de Humanidades e Tecnologias in Lisbon.

The engine was built using:

- Python 3.6
- Pygame (https://www.pygame.org/news)
- Numpy (https://numpy.org/devdocs/user/quickstart.html)
- Numpy-quaternion (https://pypi.org/project/numpy-quaternion/)

There is a sample application that can be run by using: `py.exe sample.py` or `python sample.py` or `python3.6 sample.py`, depending on your Python installation.

Assignment

The assignment for the course is as follows:

- Build a "Viewer" application. You can use the sample application as a basis. That application has to feature the following functionality:
    - Display a 3d object (see below) in the centre of the screen. Control of the visualization has to be done using the following keys:
        - Left/Right arrow: Rotate object around its Y axis
        - Up/Down arrow: Rotate object around its X axis
        - PgUp/PgDown: Rotate object around its Z axis
        - W/S: Move object up and down relative to the screen
        - A/D: Move object right and left relative to the screen
        - Q/E: Move object forward or back relative to the screen
    - Create a model other than a cube for this display. That model can be loaded from a file (in a format like JSON, etc) or can be built totally in code. The model has to include sub-objects (like in the sample the object is made of the red cube and a child green cube)
- Build a "FPS-like" application.
    - Create an environment where the player can roam using standard FPS controls. The environment can be just a series of cubes with different scales and positions
    - Implement backface culling.
        - Backface culling stops the polygons that are facing away from the camera from being renderer
        - Hint: You can use the "face normal" and a dot product to detect these cases
        - You can check this video for a more in-depth explanation: https://www.youtube.com/watch?v=ShTiQGxiZRk
    - Implement filled geometry, replacing the wireframe
        - Hint: you'll have to sort objects by distance and draw back to front)
    - Stop objects that are behind the camera from being renderered
        - You can do this per-object, or per polygon
    - Implement very simple point lighting:
        - Create a PointLight3d class and extend the Scene class so you're able to add light(s) to it
        - Implement shading based on the light:
            - Hint: Light intensity = max(0, dot(Face Normal, Incoming Light Direction))
            - Hint: Polygon Color = Light Intensity * Color

Project delivery

- Project can be made individually or with a group of up to 3 students.
- Git commit history will be analyzed to see individual work of students in the overall project
- Project has to be delivered up 20th January 2020 (midnight), and link delivered on the course's Moodle page
    - Deliverables have to include a link to the Github repo
        - If you want to use a private repository, instead of a public one, you can deliver all the files in a .zip file, **INCLUDING** the .git directory for git usage analysis
    - Only one student in the group need to turn in the project
    - Project has to include a report, in a `readme.md` file. This report has to include the work done on the project, and the individual contributions of the group.
    - Report should also include (besides the names and numbers of students), their Github account username.
    - Report has to be formated in Markdown, as taught on the November workshop.

    - Extra credit on reports that include a short postmortem, where students explain what went right with the project and what went wrong
- Grade will consider the following:
    - How much was achieved from the overal goals
        - Viewer application is considered the minimum viable delivery
    - Functionality and lack of bugs
    - Overall quality of code, including documentation
    - GIT usage throughout the project, as well as individual contributions of students

Installation of required modules

To run the sample application, you'll have to install all the used modules:

- `pip install pygame`
- `pip install numpy`
- `pip install numpy-quaternion`

Although not needed, to avoid some warnings on application startup, you can install two additional modules:

- `pip install numba`
- `pip install scipy`

If pip is not available on the command line, you can try to invoke it through the module interface on Python:

- `python -m pip install <name of package>`

There might be some issues with installing numpy and numpy-quaternion, due to a C compiler not being available in the path. If that happens, you can try download a binary version of the library (called a wheel) and install it manually.

You can download the wheels for Numpy from `https://pypi.org/project/numpy/#files`. Choose the appropriate version for your OS and Python version (cp36 for Python 3.6, cp37 for Python 3.7, etc). For example, 64-bit Windows 10 for Python 3.6 is the file `numpy-1.17.4-cp35-cp35m-win_amd64.whl`.

For numpy-quaternion, you can get the files from `https://www.lfd.uci.edu/~gohlke/pythonlibs/`. Same naming scheme is used, so the file for 64-bit Windows 10 for Python 3.6 is the file `numpy_quaternion-2019.12.12-cp36-cp36m-win_amd64.whl`.

To install a wheel manually, you just have to run the command: `pip install <wheel name>` or `python -m pip install <wheel name>` from the directory where the wheel was downloaded to.

Work on the project

We recomend building a fork of this project, and doing additional work on your repository.

- Create a copy (fork) of this repository (normally called *upstream*) in your Github account (**Fork** button in the upper right corner). The copy of the repository is usually called *origin*
- Get a local copy (on your PC) of the *origin* repository, with the comand `git clone https://github.com/<your_username>/imfj1_2019_projecto.git` (replace `<your username>` by your username in Github)

- Link the local repository with the remote *upstream* repository with the command: `git remote add upstream https://VideojogosLusofona/imfj1_2019_projecto.git`

Periodically, update your repository with changes done on the source `imfj1_2019_projecto` repo (in case bug fixes are introduced):

- Make sure you're working on the *master* branch:
  - `git checkout master`
- Download any updates on the imfj1_2019_projecto source repository by merging them with your *master* branch:
  - `git fetch upstream`
  - `git merge upstream/master`
- Upload (*push*) the changes on *upstream* to the *origin* repository:
  - `git push origin master`

Do your normal work and commit/pull/push as taught. Grade will take in account how well GIT is used throughout the project.

Licenses

Metadata

- Autor: Diogo Andrade