The Journal of Open Source Education

Elena: Open Source JavaScript Game Engine for Educators

Roman Parise[1] **and Georgios Is. Detorakis**[1]

1 Independent Researcher

DOI: 10.21105/jose.00186

Software

- Review
- Repository
- Archive

Submitted: 13 October 2022
Published: 20 December 2022

Summary

Elena is a Flask-based Python library for building chatbot-based education trivia games. Users define lessons as modules by providing an HTML-based lab manual containing the background content. Any lab manual can be treated as a detailed activity description or as standard instructional material for a course. The user also provides a `*.json` file listing all of the trivia questions for the module. The user may also write `*.js` files to describe games that may be played within the chatbot. The Elena Python library is used to organize and extend the trivia game.

Statement of need

Gamification is becoming increasingly popular in education. Although the impact of gamification on long-term student outcomes is unclear, the goal is to increase students' motivation, which may improve these outcomes (Dichev & Dicheva, 2017; Faiella & Ricciardi, 2015). One of the primary challenges plaguing the field is the need to understand how to design a game to improve particular learning outcomes. Performing such experiments is very difficult due to the difficulty of designing games. Designing multiple games and gauging how each one impacts the player's outcomes of interest are even more challenging. Thus, there is a need for a framework that enables one to develop and modify educational games rapidly.

We know of no actively-maintained game engine for accomplishing this task. We developed one called 'Elena.' Elena is used to design trivia games that run in a web browser. The trivia game follows a chatbot-based game loop. However, the game engine is extensible in that a user can easily add additional functionality, such as minigames within the chat loop, with a basic understanding of Python/JavaScript.

Story and experience of use

Elena was initially developed as part of a cloud lab service to be used as an educational tool providing easy access to a remote laboratory for students. The authors prototyped remotely-accessible temperature control systems based on thermoelectric coolers (TECs) and researched open-source tooling for designing and simulating custom instrumentation hardware (Parise & Detorakis, 2022). The goal was to couple the temperature controller with a game to make the experience appealing to students in a classroom setting. There were difficulties with developing a game that operated well when used with the remote lab

setup. However, the game evolved into a standalone trivia app and was then generalized to a framework for developing such apps.

The examples included in the repo are based on some of the experimental work conducted by the authors. For instance, two of the provided labs detail the construction of TEC-based temperature controllers and their use in gallium melting experiments. The bonus round minigame in the second temperature controller lab was originally intended to involve the actual remote hardware.

We held a brief seminar discussing the use of thermoelectric coolers in temperature control instrumentation. We had a few attendees and used the temperature controller lab examples to reinforce the attendees' understanding of the material. The session started with a presentation on temperature control work conducted by the authors. We then provided the attendees with a link to our Heroku-hosted site with the web app. We allowed the attendees to first read the lab manual on their own before we gave an overview of its content. Afterward, we addressed any questions and then proceeded to the game.

At first, the users had difficulty answering the questions but quickly achieved correct responses. While playing the game, we would address any questions related to the material, and we even played the game with them and would describe minigames as we encountered them. Overall, the attendees felt the game was an excellent addition to the talk that helped reinforce the material. Beyond feedback about the specific trivia questions, they believed it would work well for topics with simple responses but that the game still needs to be improved to handle ambiguities in more complicated topics. However, they agreed that this would be a valuable educational tool for instructors with basic Python/JavaScript knowledge.

Architecture

The engine contains two parts, the frontend, and the backend. The backend extends Python's Flask library (Grinberg, 2018). Users define game modules using the Module class. The bare minimum requirement for developing a game is (a) the lab manual, or the description prior to the game, and (b) the `*.json` file with the trivia questions. Optionally, users can add JavaScript code for a minigame module within the chat loop.

The frontend is implemented using JavaScript/CSS/HTML and the jQuery library for animations. The goal of the frontend design is to be compatible across multiple browsers. Elena is not currently intended to work on other platforms, like smartphones.

The standard game loop code is implemented on the client-side using JavaScript/jQuery. However, most additional functionality is meant to be handled on the server side. Users define new packet types and map those to functions to be called when a packet of that type is received. This is done using the Python module code. The client-side JavaScript code includes functions for formatting and sending POST/GET requests with these custom header types. These custom request types are used to extend the functionality of the trivia game.

The game loop is a simple chatbot conversation. The bot asks the user questions, and users are rewarded with points, or "kiwis," when they answer questions correctly. The game designer may add minigames in which the user can either (a) earn kiwis (or points) or (b) pay five kiwis to play the game.

Sample Results

The repo contains an example implementation. `app.py` is an example of the Flask app file and can be found in the root directory. Within that file, the most important section of code is the object definitions. This is where the user organizes the trivia game's structure.

```
# Object Definitions

# Create necessary objects for trivia game
mymod0 = Module("Temperature Controller Part I", [])
mymod1 = Module("Temperature Controller Part II", \
                [Game("Experiment0", {}, {'bonus' : bonus}),
                 Game("ArtGen0",
                      {'art' : art},
                      {'nn_art_bio_keyword' : nn_Art_bio_keyword,
                       'nn_art_art_keyword' : nn_Art_art_keyword})])
mymod2 = Module("Bacterial Culture", [])
app = eFlask([mymod0, mymod1, mymod2], DOMAIN_NAME,
             import_name=__name__)
```

Most requests will be routed through a root route:

```
@app.eroute('/', methods=['POST', 'GET'])
def root(mod):
    """
    The root route. The basic route for all incoming HTTP requests.
    @param mod: your eFlask object, app
    @return: the HTTP response, depending on the request
    """
    print("Serving index.html!")
    return mod.handle_requests()
```

The example implementation contains three trivia sets. The first two are labs that involve the construction and testing of a hotplate-based temperature controller. The other lab has to do with culturing bacterial samples. Each lab contains its own trivia set. The second temperature controller lab in particular contains some additional minigames. One of which is labeled "Experiment0," in which users guess the material of interest using known properties thereof. The other is labeled "ArtGen0" and allows users to generate neural network art related to the course material. "Experiment0" serves as a bonus round in which users can win more points than from a typical trivia question. "ArtGen0" is a minigame in which the user must spend their earned points to play.

Trivia questions are described using a JSON file of the following form:

```
{
 "Latin name for the bacterium that causes botulism?"  : "Clostridium botulinum",
 "Latin name for the bacterium that causes tetanus?"   : "Clostridium tetani",
 ...
 "Question"                                                    : "Answer"
}
```

The starting lab manual is described using HTML markup:

```
<div style="text-align: center;"><u>Bacterial Culture</u></div>
</br>
</br>
Description: Simple bacterial culturing lab.
</br>
</br>
Materials:
</br>
<ul style="list-style-type: square;">
<li>Agar plates</li>
<li>Cotton swabs (packaged and sterile)</li>
<li>Bleach</li>
<li>Plastic ziplock bags</li>
</ul>
...
```

Bibliography

Dichev, C., & Dicheva, D. (2017). Gamifying education: What is known, what is believed and what remains uncertain: A critical review. *International Journal of Educational Technology in Higher Education*. https://doi.org/10.1186/s41239-017-0042-5

Faiella, F., & Ricciardi, M. (2015). Gamification and learning: A review of issues and research. *Journal of E-Learning and Knowledge Society*, *11*, 13–21. https://doi.org/10.20368/1971-8829/1072

Grinberg, M. (2018). *Flask web development: Developing web applications with python*. "O'Reilly Media, Inc.".

Parise, R., & Detorakis, G. (2022). OpenPelt: Python framework for thermoelectric temperature control system development. *Journal of Open Source Software*, *7*, 4306. https://doi.org/10.21105/joss.04306