Bringing AI APIs into the Classroom with a JavaScript Coding Site

Mark Humphrys

School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland
Email: mark@ancientbrain.com (M.H.)
Manuscript received August 21, 2024; revised September 4, 2024; accepted October 28, 2024; published February 14, 2025

***Abstract*—With the growth of large-scale Artificial Intelligence (AI) models has come the growth of AI Application Programming Interfaces (AI APIs), where the model remains at the remote site, and is called remotely via an API. The goal of this work is to bring accessible coding for AI APIs into the classroom. This paper describes two key innovations. First, a new JavaScript coding site, "Ancient Brain", designed for education with a focus on AI. And second, the building of an extensive set of open-source JavaScript AI API programs that can be run by students in the browser and can also be edited and modified by students in the browser (on Ancient Brain). A search of coding sites struggles to find any comparable collection of AI API JavaScript programs for students in existence anywhere. While AI APIs can be called in any programming language, this paper discusses the advantages of a JavaScript-based system. We discuss issues such as public versus hidden code, API key management, API response delays and CORS issues, with consideration of how to address these in an educational environment. Future work would be to build AI API exercises for secondary school students using this starter collection.**

***Keywords*—Artificial Intelligence Application Programming Interfaces (AI APIs), coding sites, education, open source, code archives, JavaScript**

I. Introduction

Recent years have seen an explosion of large-scale Artificial Intelligence (AI) models from new companies such as OpenAI, DeepMind and Cohere, and established companies such as Meta, Google, and Microsoft. These models are considerably larger than any made before in AI. Some existing models have over 100 billion parameters [1]. GPT-4 from OpenAI is suspected to have over 1 trillion parameters [2].

While small AI models can be downloaded by users to run locally, this is not practical for these large-scale models [3]. For these, the model remains at the remote site and access is provided through a programming Application Programming Interface (API) which may be used by apps and third-party programs in general. The standard way of calling these APIs would be using HTTP POST from any programming language. There may also be some form of webpage access, such as the ChatGPT webpage [4].

API access to large models is often controlled with subscriptions, to generate a revenue stream to recover some of the massive costs involved in creating the model. There is usually some form of free API entry level for testing, with a limited amount of usage before subscription is required.

There are also more informal collections of AI models and APIs, from large and small organizations and even individuals. For example, Hugging Face [5] hosts over 500,000 AI models and has API access. Rapid API [6] hosts tens of thousands of AI and non-AI APIs.

The amazing products of generative AI, and other AI functionality, have fast become part of the culture, and educators are considering how to bring these into the classroom. Much of the modern "AI in education" research focus is on generative AI [7], in particular on generated *text*, and in particular on ChatGPT, via manual web page usage [8, 9] rather than, as in this work, coders making API calls. Note also that the student coders in this work made use of 39 different and diverse AI APIs, rather than just one (ChatGPT).

The educational environment in [10] is constructed using multiple AI APIs, with these APIs called at runtime. However, it is not aimed at the students actually *coding* the API calls and the overall environment themselves, which is the focus of this work. The students in that work are users, not coders.

The organisation of this paper is as follows. We first introduce the problem of making AI API coding accessible. This motivates the coding site, which we explain in detail. We explain how AI API programs can be created using the site. We then survey the AI API collection created by the students. We compare with related work, which suggests no similar open-source, runnable AI API collection exists.

The problem addressed in this work is how can relatively inexperienced student *coders* integrate calls to *AI APIs* into their code. This could be done to study how AI models work, to learn how to code APIs in general, or (probably the biggest use) just to make coding itself more compelling.

In "Related work" below, we survey the literature and online resources, which indicate that no diverse open-source collection of runnable AI API programs for education exists.

This work then has two key innovations: (1) the creation of a new coding site for education to (among other things) help students code for AI APIs, and: (2) the construction by students of a unique open-source AI API collection.

The latter collection is available for educators for further use. We argue this is a step forward in making AI API coding accessible at an early coding level. We show how students, even at an introductory level, can integrate the intelligence of remote AI projects into their coding creations. We argue that this is an underexplored area in education with huge future potential.

II. A Coding Site for Education

How can coding learners be helped to construct or modify programs that make calls to AI APIs? They can of course be left alone with the manual, but learning how to make a HTTP POST to an API endpoint is hardly for the beginner.

For some years we have been engaged in a project to build a shared coding environment to allow introductory coders access to advanced coding functionality (like 3D graphics, AI algorithms and AI APIs) in a way different to, and easier than, building it all from scratch. This project was originally called the "World-Wide-Mind" project and is now called "Ancient Brain". After experimentation with multiple technologies, we built a site for collective and shared JavaScript coding in the browser, which shall now be explained.

A. Ancient Brain

Our shared coding site is called "Ancient Brain". It is explained in detail in a separate paper [11]. Briefly, it is based on the following principles:

1) All programs are in JavaScript.
2) All programs *run* in the browser, with no installation of any software needed.
3) All programs are *edited* in the browser, with no installation of any software needed.
4) The main site stores all the programs. Users (e.g., students) can view collections of programs written by other users (e.g., teachers, 3rd parties, and possibly other students). Normal programs are *public*. We have an option to set them to *be hidden* (not discoverable).
5) Users can run them securely, even though they are untrusted code (e.g., by other users). Teachers can run untrusted student code. This is thanks to the mature JavaScript security model, fundamental to the Web [12]. Note also this is a *forgiving* programming environment, where browsers will try to run bad JavaScript, even at the risk of incorrectness [13].
6) Users can normally view the source code of the programs. We will allow a program to opt out of this by being *hidden* (not discoverable). We will allow a program to *partially* opt out of this by having its code *obfuscated*.
7) Users can "clone" these programs to get a personal copy to edit.
8) Users can edit their copy, run it, and save it to the server for later use, and use by others.

Ancient Brain is live at ancientbrain.com. The reader is strongly urged to simply go there now and start running programs (which are called "Worlds"). You run Worlds by clicking on them, with no install needed. Worlds can be games, AI applications, general apps, coding challenges, math demos, physics demos, quizzes, chat applications, virtual worlds, museum tours, and more.

Fig. 1 is a screenshot of the home page as of Nov 2024, showcasing a random four Worlds. The ones shown here are: (1) "Complex World", a World showing how to do Three.js graphics on Ancient Brain. (2) "3D for Kids World", a port of a simple Three.js World from a book. (3) "AI Song Generator", a World that calls two AI APIs, MusicGen from Meta and Stable Diffusion from Runway. (4) "Recognise any image", an AI World to recognize images using ML5 and MobileNet. To find and run all these Worlds, go to the site, and use the search box.

Ancient Brain is in use in multiple teaching courses at Dublin City University, at both undergraduate and (taught) postgraduate level. At time of writing, it hosts 10,840 JavaScript Worlds written by 2,358 coders. Experiments have been run on bringing it into schools, and writing a course or textbook for schools is at the top of future work. We are seeking partners to work on this.

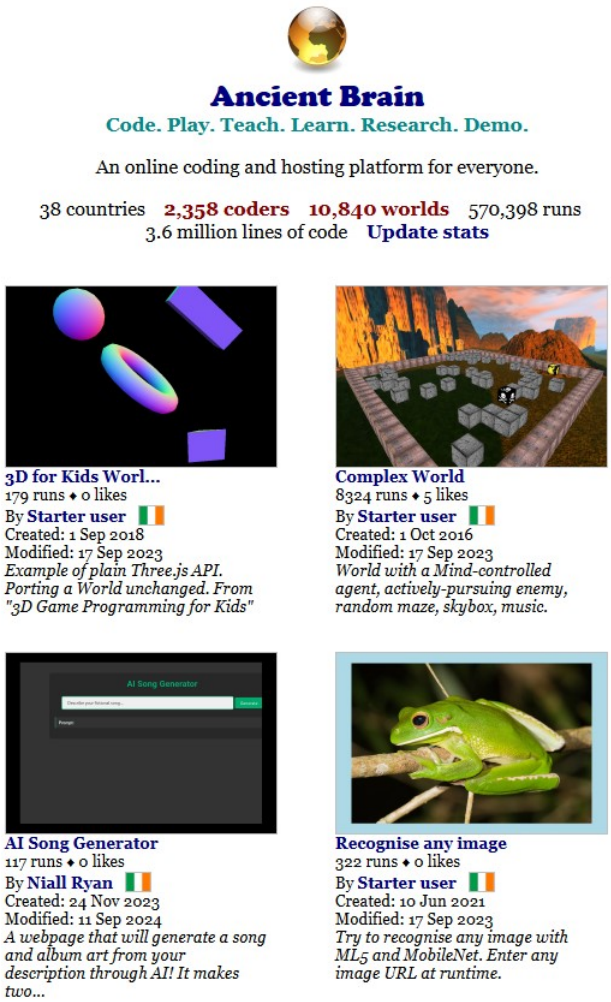

Fig. 1. Part of the home page of ancientbrain.com.

B. Ancient Brain Features

Some features of the Ancient Brain site of relevance to education and AI are:

1) An option to *obfuscate* your JavaScript code. If chosen, the public version of your code is obfuscated. When another user runs it and views the source, they only see obfuscated code. The owner of the code can see the plain text code. We have a further option *hidden URL*, where other users cannot find your code or run it.
2) "Teacher" and "Student" accounts. In a "Student" account, all code is obfuscated and at hidden URLs. Other students cannot see it. It is excluded from search and the URL is not guessable. In a "Teacher" account, you can see and run all code written by your students. You can view the plain text code, and you can *edit* the code. Teachers can easily and safely edit and even *bug-fix* student code and run it again to see what difference changes make. This we believe is a very powerful feature that makes marking a pleasure.
3) Support for AI. Support for the ML5 AI library [14]. You can upload other AI JS libraries and use them.
4) Support for 2D and 3D graphics. Built-in graphics libraries and many sample Worlds with open-source code. Support for Three.js, P5, Phaser and native WebGL

coding. We also allow the making of text-only Worlds, that are regular web pages with HTML elements.

5) Users can upload images, textures, music, sound effects, JavaScript libraries, JSON data files and 3D models to use in their Worlds. Uploads are public. You can reference and use other users' uploads in your code.
6) Support for 3D models. Built-in libraries for embedding 3D models (e.g., houses, people) in Worlds. Sample Worlds are provided. See "3D models" in the Ancient Brain docs [15].
7) Support for physics in 3D Worlds. Built-in physics libraries with gravity, momentum, etc. For example, ammo.js [16].
8) WebSocket support for all Worlds. This allows real-time communication between users running your World. Users can write multi-user Web games and real-time chat. See "Websockets" in the Ancient Brain docs [15].
9) Search engine for Worlds, users, and uploads. *Code search* to search the JS code of all public programs.

For more features, and a more detailed introduction to Ancient Brain, see [11]. Space does not allow here for screenshots of the editor and other screens. Just to note that school children with no coding experience have been able to code on this site in their first lesson. We recommend the new user goes immediately to one of the "starter tutorials" [17, 18] to start coding on the site.

For an overview of the extraordinary range of *non*-AI programs that users have created on the site, see [19].

III. AI APIs in JavaScript

Given this JavaScript coding platform, how do we call AI APIs? The standard way of calling AI APIs is by using a HTTP POST request, providing input data to the POST. Some simpler APIs use HTTP GET, with the data in the URL. A HTTP request can of course be made from any normal programming language.

The JavaScript needed can be quite complex. For instance, the following will make a call to the GPT API at OpenAI [20]. Not shown are: (1) setting up the prompt and other data in 'inputdata', (2) setting up Ajax headers to use the API key, (3) defining the success function to handle the return data whenever it comes back, and: (4) error handling. For the full code see "Chat with GPT model" [21]. The call to the API is made here using jQuery:

```
$.ajax({
    type: "POST",
    url: "https://api.openai.com/v1/chat/completions",
    data: inputdata,
    dataType: "json",
    success: function ( data, rc ) { ... }
 });
```

A note is useful here on the concept of JavaScript calling an API from the browser. The "same origin policy" is the core idea of web app security that JavaScript should only make server requests ("Ajax") to the server it came from. To get around this, an API endpoint will use Cross-Origin Resource Sharing (CORS) [22, 23] to allow it to be called from any JavaScript. This can be seen in the browser debug tools, where, when you run the above World on Ancient Brain, you can see the HTTP response headers include `Access-Control-Allow-Origin: https://run.ancientbrain.com`. This means the original must allow *all* origins. A normal site would have no Access-Control-Allow-Origin response header at all, and could not be called from our JavaScript.

Is the coding learner expected to write from scratch complex HTTP POST requests as above? Not really. We expect them to run *existing Worlds* that make these API calls, clone these Worlds, and *partially* modify them, probably without ever going near the HTTP POST section.

In fact, Ancient Brain now has a built-in function `AB.callGPT` to call the GPT API in a much simpler way, replacing all the complexity in the box above, plus the not-shown parts (1), (2) and (4). At any point in the student's code, they can simply call GPT as follows. Define the API key and prompt, and then:

```
AB.callGPT ( apikey, prompt, function ( reply )
{
   // do something with the reply
});
```

This is easy enough that an introductory coder could do it. See "APIs and CORS" in the Ancient Brain docs [15]. Future work would include building these kind of Ancient Brain wrapper functions to call other APIs.

IV. An Open-Source AI API Collection

The second part of this work was the building of a diverse open-source AI API collection for students to run and edit. Three classes of undergraduate and taught postgraduate Computer Science students were introduced to JavaScript AI coding on Ancient Brain, and then set to work (in late 2023) to write JavaScript "Worlds" to call AI APIs of their choice to demonstrate their power. Table 1 shows the APIs that were chosen by the students (total of 39 different APIs).

Table 1. APIs chosen by the students

API	Number of Worlds
OpenAI GPT	73
OpenAI DALL-E	24
Rapid API (14 different APIs)	13
Cohere	4
OpenAI Whisper	4
Hugging Face (4 different APIs)	4
Stable Diffusion	3
D-ID	2
OpenAI TTS	2
Microsoft Azure Vision	2
Deepgram	2
Deep AI	2
Elevenlabs	2
OpenAI Assistants	1

Google Cloud Translation	1
Google Cloud Vision	1
Google TTS	1
Ollama	1
Replicate	1
Eden AI	1
Twinword	1
Wikicheck	1
Clipdrop	1

V. Example Worlds

Some of the most impressive Worlds created by students in this experiment are as follows. To see and run these, go to "AI API Worlds" in "Showcase Worlds" [24].

A. Non-Graphics Worlds

We start with Worlds that are basically webpages, rather than 2D or 3D graphics environments.

Infinite story book [Fig. 2]. This World generates an illustrated story book using prompting. It calls the GPT API for text and the DALL-E API [25] for images. The user enters a few prompts and GPT constructs an entire story. Then we call DALL-E with that story to get images to represent it. You can go back and forward, and the text and images are still there. Author: Darragh McGonigle.

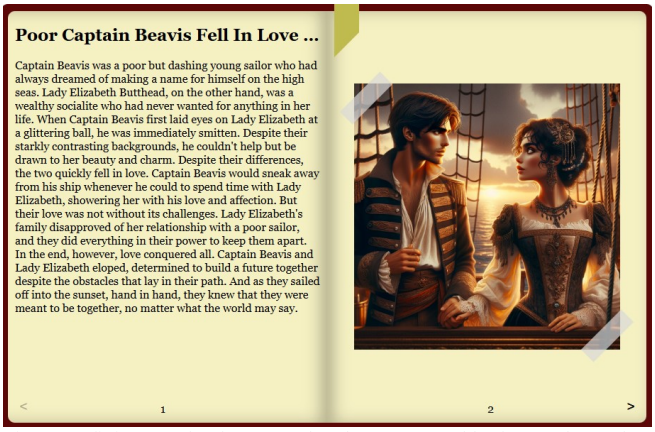

Fig. 2. "Infinite story book". From the Showcase Worlds above. The student gets GPT to make a story and DALL-E to make images.

Therapeutic AI [Fig. 3]. Therapeutic AI bot. The user says how they are feeling. A GPT API call generates a text response. This text is given to the D-ID API [26] (text to video clips with photo-realistic characters speaking). The photo-realistic character then speaks the advice that came back from GPT. Author: Moses Crasto.

Translation. Chat to someone in a different language. This World uses the Ancient Brain Websockets service to make an AI-powered chat app. Two users in different languages can talk to each other and the system translates. Users run the World at the same time and select languages. When they type, it calls Google Cloud Translation API [27]. Authors: Renso Guilalas and Christopher Muthi.

DEB.ai.TE. An app for school and university debates. The user enters a prompt, such as a controversial statement for a debate. The app makes two calls to the GPT API to get opinions, one for and one against. In the return page, the user selecting some text makes a call to the WikiCheck fact-checking API [28]. Also, every word is clickable, and a click makes a call to the Free Dictionary API to get the word definition. Authors: Calem McGlynn and Joseph McNaney.

Generate Images from Audio. Audio to text to image. The user needs to turn the microphone on for the Ancient Brain site. The user records some speech, and can play it back. The user then sends the audio to the OpenAI Whisper transcription API. This returns a transcript of the audio. The transcript is sent to DALL-E to generate an image. Author: Toma Emoghene-Ijatomi.

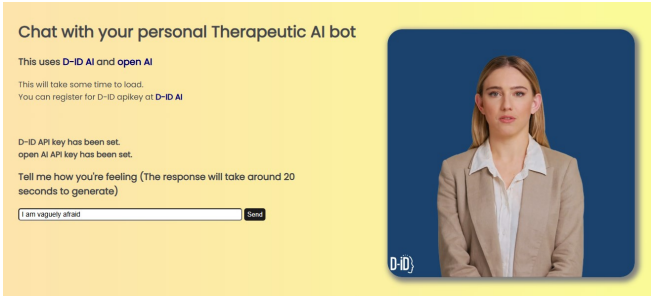

Fig. 3. "Therapeutic AI". The student gets D-ID to make video clips with photo-realistic characters speaking.

B. Graphics Worlds

The previous Worlds are regular web pages that call the APIs and put HTML elements on the page. The power of the Ancient Brain platform can be seen in the following projects that combined AI API calls with 3D graphics Worlds. Each of these projects created a compelling 3D graphics World and then used AI API calls in the background to populate it, drive gameplay, and so on. Again, see [24] to run these (Fig. 4).

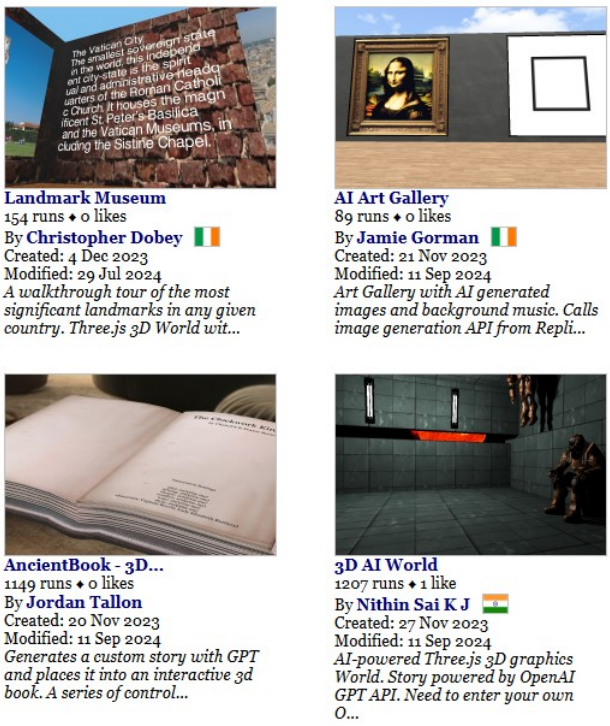

Fig. 4. Some of the graphical Showcase Worlds at [24]. Go to that page and click on any of them to run them.

AI Art Gallery [Fig. 5]. Dynamic 3D art gallery with AI generated images. This is a Three.js World. The user can rotate the camera. The user can generate art for each frame in the gallery. This calls the Replicate image generation API [29]. For example, ask for "Mona Lisa", or invent your own. It paints the new image on the wall. Author: Jamie Gorman.

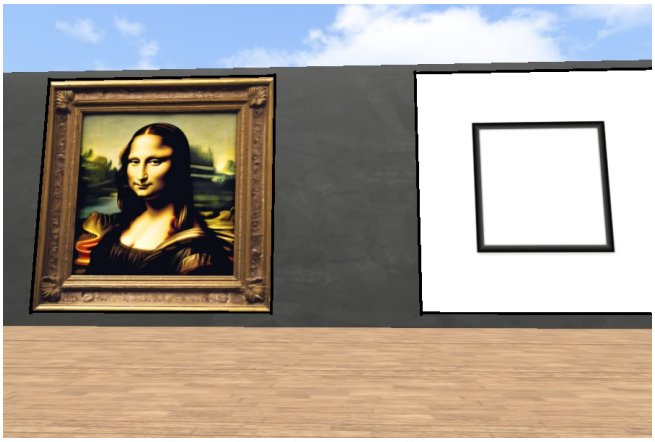

Fig. 5. "AI Art Gallery". The student gets Replicate to generate pictures to populate the frames.

AncientBook [Fig. 6]. This generates a custom story in an interactive 3D book. Controls on the page help the user design the story. Then OpenAI GPT generates the story. The story is inserted into a 3D model of a book in a Three.js World. The user can scroll to change pages. There is a "page turn" sound effect. The user can drag the camera, rotate the book, and see the cover and surroundings. Author: Jordan Tallon.

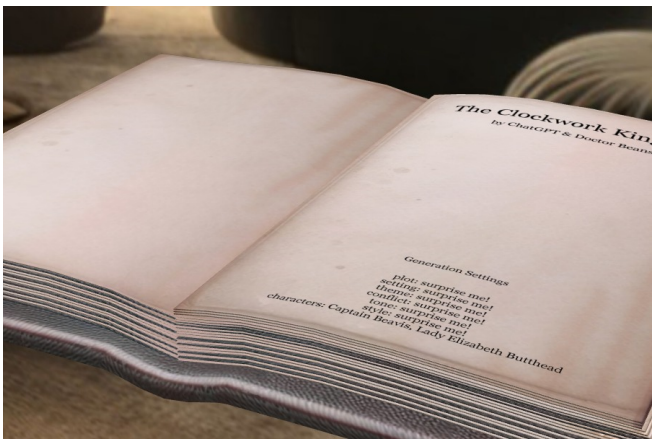

Fig. 6. "AncientBook". The student gets GPT to make a story and inserts it into a 3D model.

3D AI World. 3D game with GPT gameplay and audio. This is a Three.js World. It has 3D models with moving parts. The user can control the camera. The story is powered by GPT API. The user types, and then audio of the text comes out. The AI call then responds in a text format, and then responds in audio. Author: Nithin Sai K. J.

Landmark Museum. Dynamic 3D museum of landmarks in a chosen country. This is a Three.js World with mouse and WASD camera control. The user enters a country. The code calls the OpenAI GPT API to get a landmarks list for that country. The code then displays this as text on walls inside the World. It then calls Google Image Search API to get images of the landmarks. It displays the images on the walls. Author: Christopher Dobey.

Gatekeeper. 3D game with GPT gameplay and audio. This is a Three.js World with 3D models. The user navigates using the keyboard. If you get close to the wizard, you are detected, and he speaks. It uses the GPT API to generate a set of riddles to open the gate. The wizard's speech text is converted to audio with the Google TTS (text-to-speech) API. The user types and user replies are also converted to audio. There is background music and sound effects. There is a Three.js camera that the user can rotate. Author: Sam Murphy.

VI. Issues in AI API Coding

A. API Key Management

One issue that arose with this project and will arise in general with using APIs for education, is API key management. Most APIs require registration for an API key, which is basically a password. There is often a free entry level and then paid subscriptions. APIs normally ask users to keep keys secret, not least because other users seeing your key means they can use it, and use up your credits.

In an educational setting, should the teacher register a key, and then copy it to the class? The class might soon exhaust the teacher's credits. Or should students have to register themselves, which imposes extra hurdles for them?

One obvious issue arises with JavaScript, which is that the JavaScript is client-side, meaning if you put the API key in your code, then anyone running your World can view source and see the API key [30]. Solutions to this could be:

1) Keep the World "Hidden URL", so that no one can see the key except trusted people.
2) Require the user to enter an API key when they run your World. This is what "Chat with GPT model" [21] does.

B. API Response Delays

One downside of the AI being remote rather than local is *responsiveness*. Delays of seconds in response time are seen in some APIs with some subscriptions - notably APIs from small organizations and free entry level. On the other hand, APIs with large infrastructure behind them, and using a paid subscription, often show remarkably fast response times.

From the student point of view, the issue with an API call is that the answer will come back at some point in the future. This is *asynchronous* JavaScript, and coding for it is a skill that teachers and students must learn. For a complex series of API calls returning at different times, see the source code and console output of "Student Grading Assistant" [31].

C. CORS Return Data

For some of the above APIs, issues arise with the return data. The data may be returned directly with the Ajax call, in which case there is no problem.

A more difficult question (and this is the norm with some APIs) is when the return data is the URL of a resource on a normal (non-CORS) web server. How can the JavaScript fetch that resource? Recall that the "same origin policy" means resources we fetch should be on the same server as the JavaScript (i.e. Ancient Brain). However, we do not have the return data uploaded to Ancient Brain, since we did not know

about it in advance. It only appears at runtime.

For such return data, we can use a *CORS proxy* like corsproxy.io [32] to fetch arbitrary resources off the Internet. In this case a server-side program at the CORS proxy fetches the resource, outside of the browser environment, which means no "same orgin policy" is needed or enforced. The CORS proxy then serves up the resource using a CORS header and so any third-party JavaScript can fetch it.

The code looks like:

```
var url  = ...   // some resource on a normal website

var curl = "https://corsproxy.io/?"
  + encodeURIComponent ( url );   // CORS-enabled URL

$.get ( curl, function ( data )
{
  // we have got the data
});
```

We can use this remote CORS-enabled URL in other places where normally one expects a local same-server resource, such as loading textures into graphics Worlds. For example, the DALL-E API outputs AI-generated images to a server under blob.core.windows.net, which is then non-CORS. They cannot be directly fetched into graphics Worlds, but using a CORS proxy they can.

Ancient Brain now has a handy function `AB.cors` to provide a CORS-enabled version of the URL. See "APIs and CORS" in the Ancient Brain docs [15]. So, grabbing a random image from the Internet to put in a P5 graphics World looks like this:

```
img = loadImage ( url );             // fails
img = loadImage ( AB.cors (url) );   // works
```

This is easy enough that an introductory coder could do it. It should be noted that there are lots of issues with CORS proxies [33], including that the proxy might vanish next year.

VII. Empirical Results and Related Work

173 students took part in the project, which was part of their course and for which they received marks, so they were motivated, and we were not dependent on self-selected volunteers. We consider the results under a number of performance headings as follows.

A. Learning Curve

Most students before the project had little JavaScript experience, no experience coding for APIs or AI in JavaScript, and no AI API experience in any language. That statement is based on comments in the student documents, but we do not have exact numbers. In the next run of the project, we will gather exact numbers on what experience the students have before we start.

B. RAM, Disk and Other Requirements

No memory usage or speed issues were reported with client-side requirements for the site. Students were free to use their own PCs, laptops and mobile devices. The demands on their machines by the site are no worse than the demands of normal web usage. Coding and running 3D graphics from the site are possible on any modern smartphone, and on any laptop or PC built after about 2015 or so.

There are no disk usage issues since storage is on the server.

C. Student Output

In a five-week period, the students submitted 211 final JavaScript Worlds for marking, of which we described only a sample above. Each student typically had also coded half a dozen practice Worlds, making probably over a thousand Worlds generated in the experiment.

D. Related work: Coding Sites

Many JavaScript and non-JavaScript coding courses for education exist. Ones using JavaScript tend to use some form of online editor, though some are entirely offline. See [34] for a JavaScript course for high school that uses an online editor at tutorialspoint.com [35].

See [11] for a detailed comparison of Ancient Brain with other coding sites. We will not repeat that here, but note the conclusion that many features of Ancient Brain are minority or even *unique* in the world of coding sites, namely:

- Search box for all user programs.
- Search box for all source code of all user programs.
- Ability to obfuscate source code.
- Teacher and student accounts where student code is hidden but teacher can view, run, and edit it.
- Users can upload images, music, sound effects, JavaScript libraries, JSON data and 3D models
- Users can browse each other's uploads.
- Programs can use other users' uploads.
- Websockets server to allow coding of real-time Websockets programs.

As discussed further in [11] this set of features makes Ancient Brain novel and different to existing coding sites.

E. Related work: AI APIs on Coding Sites

Now we look specifically at AI API programs on other coding sites. A lengthy search of coding sites struggles to find any comparable collection of AI API JavaScript programs for students in existence anywhere. The following is a survey of what does exist.

See [36] for a 2024 survey of AI courses in schools (K-12), including some *visual* programming systems (i.e., like Snap! or Scratch) that use AI APIs. A 2021 survey of Machine Learning in K-12 is in [37], which mostly lists visual programming systems and no AI API system. We have not been able to find an experiment with AI APIs in regular *text* programming (like JavaScript) for schools, but some surely exists.

JavaScript to call AI APIs is new enough, even outside schools. A search of Codepen [38] for JavaScript "pens" that call the GPT API finds various experiments, but one struggles to find any working one where it is clear what it does and one can enter an API key to talk to the GPT API. Some of them struggle with obsolete endpoints - likely to be an ongoing problem with API programs into the future. Others suffer from not explaining what they do - which can be a problem on all user-submitted sites, including Ancient Brain. The only working Codepen example found on a lengthy search was "AnalyseThis" [39].

After a search, we discover that the P5 editor [40] hosts some user-generated JavaScript "sketches" that call the GPT

API, but a lengthy search struggled to find a single working one. Some struggled with obsolete endpoints and obsolete models. P5 sketches and features for education are surveyed and discussed in some recent papers: [41] and [42], but neither of these mention AI.

JSFiddle [43] has some GPT API experiments, but we were unable to find a working one. CodeSandbox [44] has some GPT API projects but a lengthy search struggled to find one that was both public and runs.

A search of Ancient Brain for "GPT" finds pages of working GPT Worlds, though admittedly helped by curation of results after the 2023 project.

VIII. Conclusion

We believe the resulting open source code archive is impressive and relatively unique, but whether this is the best way of helping a class code AI API programs is hard to say, since there is so little similar work.

From the educational viewpoint, AI APIs provide the user a far more flexible form of access to remote large-scale machine intelligence than human-oriented web page access. Of course, the user must be a coder. Being able to bring this intelligence inside your own program is a new frontier of educational coding, and one that should have massive appeal to students.

These amazing Worlds were written by Computer Science students. Do we expect introductory-level students, such as at school (K-12) level, to be able to write such Worlds from scratch? Of course not. The value of this experiment is:

1) It demonstrates what can be done with Comp. Sci. students on the platform.
2) It demonstrates what can be done *at all* on the platform. For some of these Worlds, it was unclear whether they could even *exist* until they were written.
3) It is the basis for *exercises* for introductory-level students, where they could be tasked to clone and modify these programs in certain ways.
4) It is the basis for much *simpler* versions of these Worlds for introductory-level students to edit. In fact, we have such Worlds. See the simple "Hello World" versions, under "AI API Worlds" in [24].

For reasons given above, we have only tested students coding for AI APIs in *JavaScript*, in a browser-based environment. However, many of the issues, such as API key privacy and asynchronous return, will be applicable to other coding environments. The CORS issue will be applicable to other browser environments.

Ancient Brain does not yet have built-in support for languages that *transpile* to JavaScript, such as TypeScript and others. Nor does it yet have built-in support for AI generation of code from *natural language*, though the AI API programs here show how that might be done. In general, the user interface is written in English and there is not yet language support for other languages.

The diversity of APIs called, and Worlds made by the students in one experiment is extraordinary. All the above Worlds are open source and can be "cloned" and edited in the browser on Ancient Brain. Simpler versions could be made of them for classes with other students. Students could be set exercises to modify them in different ways. Future work is to design a course to do that.

To summarise, a limitation of this work is that we have not tested AI API coding on the site with *introductory-level* coders. This will be the focus of future work.

We are actively seeking a partner to work on an "AI for schools" book using JavaScript on Ancient Brain. Apart from the AI API Worlds shown above, the site also has a course in AI concepts at [45] (Fig. 7). This is an introduction to AI concepts, with examples including A-star search, genetic algorithms, neural networks, backprop and image recognition. These are all coded as JavaScript Worlds on Ancient Brain that can be run, cloned, and edited. We have also made student exercises. This AI code runs locally, not remotely like the APIs. We are seeking a partner to translate this, and the AI API work, into a book and course for schools and colleges internationally.

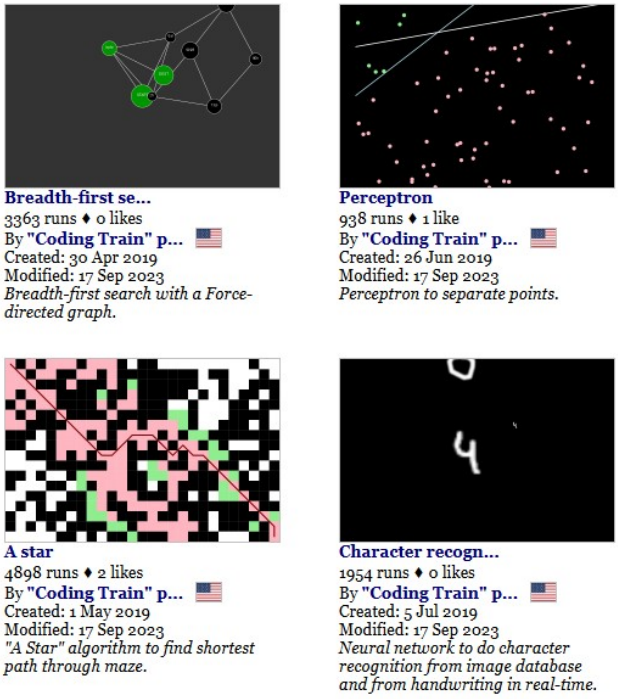

Fig. 7. Some of the Worlds used in the AI course on Ancient Brain [45]. Shown here are JavaScript Worlds for breadth-first search, perceptron, A-star search, and multi-layer neural network for character recognition. Each comes with student exercises. We are seeking partners to help turn this into a book.

Conflict of Interest

The author declares no conflict of interest.

References

[1] T. B. Brown *et al*., "Language models are few-shot learners," in *Proc. the 34th International Conference on Neural Information Processing Systems*, December 2020, pp. 1877–1901.

[2] M. Schreiner (2023). GPT-4 architecture, datasets, costs and more leaked. *The Decoder*. [Online]. Available: https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

[3] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Computing Surveys*, vol. 55, issue 12, pp. 1–37, 2023.

[4] OpenAI. ChatGPT (webpage access to GPT model). [Online]. Available: https://chat.openai.com/

[5] AI models list. *Hugging Face*. [Online]. Available: https://huggingface.co/models
[6] Rapid API Hub. [Online]. Available: https://rapidapi.com/hub
[7] A. Barrett and A. Pack, "Not quite eye to A.I.: Student and teacher perspectives on the use of generative artificial intelligence in the writing process," *Int. J. Educ. Technol. High. Educ*., vol. 20, p. 59, 2023.
[8] C. B. Fontao *et al*., "ChatGPT's role in the education system: Insights from the future secondary teachers," *International Journal of Information and Education Technology*, vol. 14, no. 8, pp. 1035–1043, 2024.
[9] K. Fuchs and V. Aguilos, "Integrating artificial intelligence in higher education: Empirical insights from students about using ChatGPT," *International Journal of Information and Education Technology*, vol. 13, no. 9, pp. 1365–1371, 2023.
[10] S. Banjade, H. Patel, and S. Pokhrel, "Empowering education by developing and evaluating generative AI-powered tutoring system for enhanced student learning," *Journal of Artificial Intelligence and Capsule Networks* 6, no. 3, 2024, pp. 278–298.
[11] M. Humphrys, "Ancient brain: A JavaScript coding platform for education with 3D graphics, Websockets, AI and support for teachers," presented at 8th International Conference on Digital Technology in Education (ICDTE 2024), Berlin, Germany, Aug. 7–9, 2024.
[12] M. Vervier *et al*. (2017,), Browser Security WhitePaper, X41 D-SEC GmbH, Aachen, Germany. [Online]. Available: https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf
[13] A. Younang and L. Lu, "Static checking of range assertions in JavaScript programs," *International Journal of Computer Theory and Engineering* vol. 9, no. 5, pp. 346–350, 2017.
[14] ML5 AI library (multiple authors). ML5 project. [Online]. Available: https://github.com/ml5js/ml5-library
[15] Ancient Brain docs. Ancient Brain. [Online]. Available: https://ancientbrain.com/docs.php
[16] A. Zakai *et al*. ammo.js physics library. [Online]. Available: https://github.com/kripken/ammo.js
[17] P5 Starter Tutorial. Ancient Brain. [Online]. Available: https://ancientbrain.com/p5.start.php
[18] Three.js Starter Tutorial. Ancient Brain. [Online]. Available: https://ancientbrain.com/three.start.php
[19] Editor's Choice Worlds. Ancient Brain. [Online]. Available: https://ancientbrain.com/worlds.choice.php
[20] OpenAI developer platform. [Online]. Available: platform.openai.com
[21] Chat with GPT model. Ancient Brain. [Online]. Available: https://ancientbrain.com/world.php?world=2850716357
[22] Cross-Origin Resource Sharing (CORS). MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
[23] G. Meiser, P. Laperdrix, and B. Stock, "Careful who you trust: Studying the pitfalls of cross-origin communication," in *Proc. the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 110–122.
[24] Showcase Worlds. Ancient Brain. [Online]. Available: https://ancientbrain.com/showcase.php
[25] DALL-E API at OpenAI. [Online]. Available: https://platform.openai.com/docs/guides/images
[26] D-ID API. [Online]. Available: d-id.com
[27] Google Cloud Translation API. [Online]. Available: https://cloud.google.com/translate/docs/basic/translating-text
[28] WikiCheck fact-checking API. [Online]. Available: https://github.com/trokhymovych/WikiCheck
[29] Replicate API. [Online]. Available: replicate.com
[30] H. K. Lu, "Keeping your API keys in a safe," in *Proc. 2014 IEEE 7th International Conference on Cloud Computing*, pp. 962–965.
[31] D. Stirbys. Student grading assistant. Ancient Brain. [Online]. Available: https://ancientbrain.com/world.php?world=5696831766
[32] The corsproxy.io CORS proxy service. [Online]. Available: corsproxy.io
[33] T. Perry. What are CORS proxies, and when are they safe? HTTP Toolkit. [Online]. Available: https://httptoolkit.com/blog/cors-proxies/
[34] C.-Y. Huang and M. Bachrach, "A JavaScript curriculum for high school students to explore computer science," *International Journal of Information and Education Technology*, vol. 8, no. 12, pp. 848–854, 2018.
[35] Online Javascript Editor at tutorialspoint.com. [Online]. Available: https://www.tutorialspoint.com/online_javascript_editor.php
[36] S. Grover, "Teaching AI to K-12 learners: Lessons, issues, and guidance," presented at the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, USA.
[37] I. T. Sanusi *et al*., "Survey of resources for introducing machine learning in K-12 context," *IEEE Frontiers in Education Conference (FIE 2021)*, Lincoln, NE, USA.
[38] Codepen. [Online]. Available: codepen.io
[39] "AnalyseThis" on Codepen. [Online]. Available: https://codepen.io/bryanhurley2/ pen/xxmgNoV
[40] P5 editor. [Online]. Available: editor.p5js.org
[41] T. Terroso and M. Pinto, "Programming for non-programmers: An approach using creative coding in higher education," presented at Third International Computer Programming Education Conference (ICPEC 2022), Schloss-Dagstuhl-Leibniz Zentrum für Informatik.
[42] A. M. McNutt, A. Outkine, and R. Chugh, "A study of editor features in a creative coding classroom," in *Proc. the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15.
[43] JSFiddle. [Online]. Available: https://jsfiddle.net/
[44] CodeSandbox. [Online]. Available: https://codesandbox.io/
[45] AI programming exercises. Ancient Brain. [Online]. Available: https://ancientbrain.com/course.ai.php