The Journal of Open Source Software

An easy-to-use p5.js 3D object picker for visual artists

David J Harris[1]

1 Queensland College of Art, Griffith University

DOI: 10.21105/joss.00475

Software

- Review
- Repository
- Archive

Submitted: 16 November 2017
Published: 04 December 2017

Summary

Visual artists who are relatively new to using code as a tool for developing new artistic ideas and processes have programming languages such as Processing (Fry and Reas, n.d.) and p5.js (Mccarthy, n.d.) as accessible entry points. However, artists who wish to start exploring more complex visual environments, such as interactive 3D models, face a steep learning curve as these applications are not built in to those languages nor covered in the standard references (Mccarthy, n.d.; McCarthy 2016).

Existing (lack of) 3D object picking in p5.js

p5.js does not have any built-in capacity to identify elements of a 3D model from the position on a 2D canvas on which it is projected. That means that developers don't have an easy way to make applications in which a user is able to interact with the 3D model via a 2D canvas by position, mouse click, or equivalent.

To make the identification between 3D model and 2D canvas in a conventional approach would require relatively complex calculations based on knowledge of camera position, location in 3D space of objects, and detailed geometry of 3D elements. For example, identifying whether a point on the 2D canvas is "on" a 3D torus primitive becomes a difficult geometrical calculation if one is to avoid false identifications with the torus hole. Furthermore, there is no built-in functionality to determine the camera position from a canvas object.

Principle of how mPicker works

This set of functions employs a workaround by using a parallel hidden canvas that draws a copy of each 3D primitive in a color that is defined by the primitive's identifier (a user-chosen integer ID that is used as a color value). Then objects are identified by simply examining the color of the pixel on the hidden canvas corresponding to a mouse click or other decision technique and returning the color value as object identifier.

More complex objects can be constructed by using the same object ID when creating each primitive. Then picking any point on that compound object will return the same ID.

User case

This set of functions provides an easy-to-use entry point for visual artists who are prototyping visual concepts for artistic application, whether the outputs are eventual browser

implementation, or as proof-of-principle for virtual reality or other interactive 3D applications.

As many artists new to coding are not yet familiar with object-oriented programming and begin with simple functional programming, the mPicker code consists of a set of functions that allow artists to replace standard p5.js 3D functions with consistently named variants, requiring just a single extra parameter for the generation of primitives. Then simple functions allow them to identify objects based on interaction with the 2D canvas on which the 3D model is projected. This approach allows artists to explore new ways of creating artistic output without needing to first build a more extensive background of coding skills and without needing to perform any mathematical calculations.

The set of functions can either be included as a javascript file or simply copied into a p5.js sketch.js file for the simplest possible use.

This module was developed by the author as part of graduate work at the Queensland College of Art, Griffith University, and tested with undergraduate students in the course "Creative Coding".

References

Fry, Ben, and Casey Reas. n.d. "Processing.org." *Processing.org*. https://processing.org/.

McCarthy, Lauren. 2016. *Getting Started with P5.js : Making Interactive Graphics in Javascript and Processing*. San Francisco, CA: Maker Media, Inc.

Mccarthy, Lauren. n.d. "P5.js." *P5.js*. https://p5js.org/.