RESEARCH ARCADE

DEVELOPMENT GUIDE

20
22

Written by Benjamin Williams
June 2022

Table of Contents

System Information

Software

There are three pieces of software running on the arcade machines at any given time:

- A joypad -> keyboard mapper which a) translates joypad inputs into keystrokes and b) assigns virtual indices to the two joypads to avoid troubles with USB enumeration.*
- Google Chrome, executed in kiosk, fullscreen and incognito mode. This is what the games/arcade software runs inside.
- A daemon which boots the above software when needed.

All games which are hosted on the machines are built with HTML5 & WebGL, meaning you will need to target this platform when developing your game. Luckily, the majority of existing game engines feature some type of build process to WebGL. With that being said, it may be worth testing on your machine to ensure that no functionality has been dropped through the WebGL build. As an example, you may encounter some issues with shaders during compilation, especially if you're using features which GLSL ES does not support.

* More information on keyboard mappings/inputs can be found later in this document.

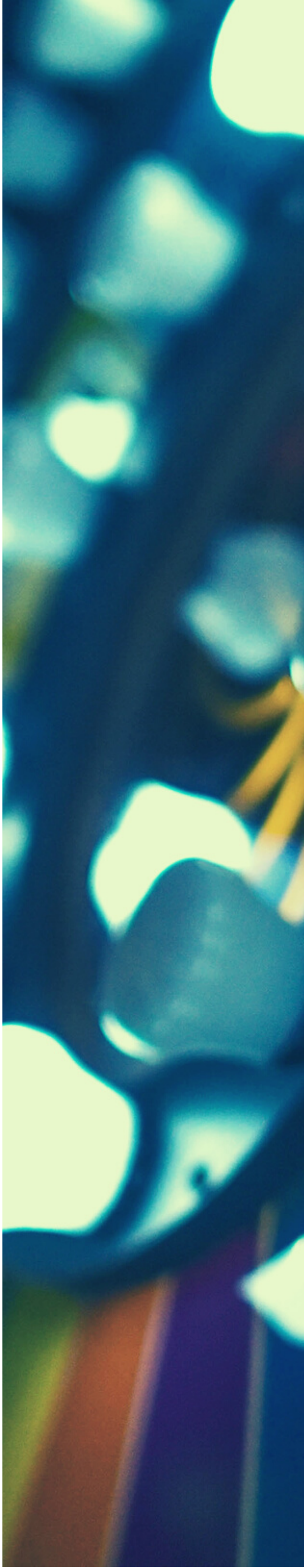
Hardware

The machines inside the arcade cabinet are of a fairly low hardware spec. It is worth taking this into account when developing your game, as performance may be an issue if your game is intensive. As a result, we suggest to test your game on a slower, older computer to ensure that it runs smoothly.

Another thing to consider is the resolution of the monitors inside the arcade cabinets. The cabinets feature a 1920x1080 resolution, and all games will be played in fullscreen mode within the browser. If your game is developed without this in mind, some visual artefacts may be encountered -- such as misalignment of UI elements.

Platform limitations

As the arcade software is primarily web-based, you can't do a few things. One of these is the majority of system-level calls, such as the ability to create, modify or delete files. Instead, you will have to either use web-based database approaches (such as local storage, or services such as Firebase) or make use of functions dedicated to building for web, within your engine. For a more comprehensive list, it is probably worth looking up the design considerations for web within the documentation of your engine. This will probably highlight some approaches to circumvent these problems.



Required Functionality

Requirements for developers

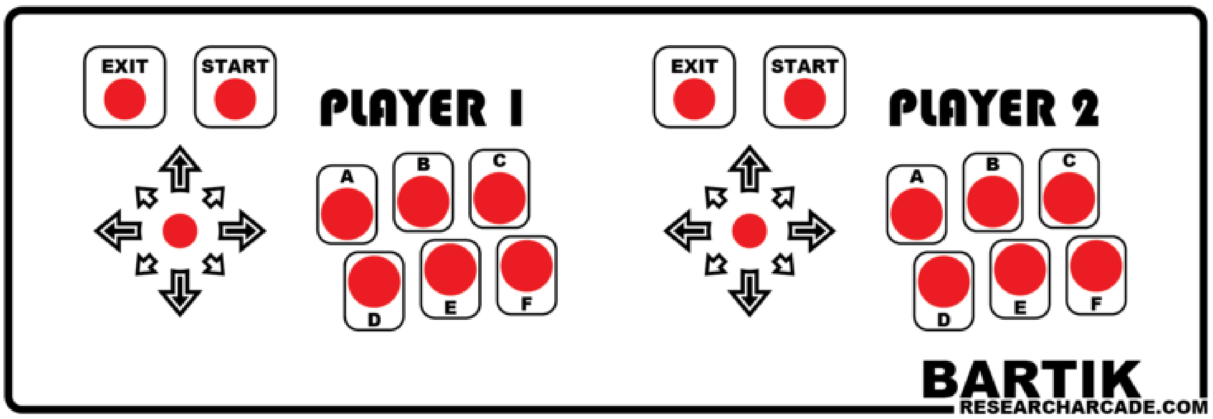
You are required to implement the following features into your game before it can be successfully deployed to the arcade machines:

- The ability to exit out of the game, at any point, if the exit button is pressed.The game will exit to the main menu screen if no key presses are detected after 30 seconds. More info on how to detect if the exit button is pressed is discussed later.
- Using **exit()**, **Application.Quit()** or other native exit functions will not work in web builds. Instead your game should access a URL in the browser to exit the game. This is possible in most game engines (for example, in Unity, you could use **Application.OpenURL(url)**).
- A reasonable description of how the game is played. A main menu prior to the main game, which should feature at least a Start and Exit button.
- All menus/selectable elements must be selectable via key presses, as the mouse is not present in the current arcade set up.
- The keyboard mapper works by detecting joypad inputs, and calling native keyboard events. Please ensure that you do not use joypad inputs in your game, as this will trigger two events (one for joypad inputs, one for keys). Please just use keyboard events.

Keyboard mappings

Physical button layout

Below is an image showing the physical button layout for the arcade cabinets. It is worth noting that this layout is identical regardless of which machine is targeted. For virtual keyboard mappings, please see the next page, in which they are listed in full.



Virtual key mappings

The virtual keyboard mappings are shown below for both players. These maps show how physical button presses are translated to virtual key presses. For example, if you need to tell if the player has pressed the Exit button, you would check if the Q key is pressed.

Button	P1 Key Press	P2 Key Press
Exit	Q	Q
Start	Return	Backspace
A	Z	F
B	X	G
C	C	H
D	V	J
E	B	K
F	N	L
Up	Up arrow	W
Down	Down arrow	S
Left	Left arrow	A
Right	Right arrow	D

Research Arcade: Unity SDK

About the SDK

The Unity SDK for the Research Arcade platform is a small package you can import directly into your Unity project to migrate to the Research Arcade platform. The SDK mainly provides syntactic sugar for detecting inputs, and methods for navigation. Further details are provided in the repository itself, which can be found below.

https://github.com/uol-arcade/research-arcade-unity-sdk

Troubleshooting

Get in touch

Are you having problems/issues developing your game for the arcade machines? Do you have any questions relating to this document? If you need help, the best idea is to get in touch with us. Contact details can be found below.

RESEARCH ARCADE

Email
help@socstech.support

Office
INB1201