

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies



Bachelor Thesis

Game Development Using HTML5

Viktoriiia Pukha

© 2018 CULS Prague

BACHELOR THESIS ASSIGNMENT

Viktoriiia Pukha

Informatics

Thesis title

Game development using HTML5

Objectives of thesis

The work is dedicated to the development of a 2D game using HTML5. The main goal is to introduce the process of web game development with HTML5, Javascript, and Phaser framework. The subgoal is to design and implementation of a platform game with selected technologies.

Methodology

The work consists of a brief analysis of technologies that can be used during 2D game development. Comparison of the main frameworks is introduced.

Implementation part describes a process of web game development, using technologies that were described in theoretical part. JavaScript and Phaser framework is used to create a platform game.

The proposed extent of the thesis

30-40 pages

Keywords

HTML5 JavaScript Phaser canvas web game 2D

Recommended information sources

Dr. Stephen Gose Phaser.js Game Design Workbook: Game development guide using Phaser JavaScript Game Framework, 2017

Geary, David M. Core HTML5 canvas : graphics, animation, and game development, 2012, ISBN 978-0-13-276161-1

Jacob Seidelin HTML5 Games CREATING FUN WITH HTML5, CSS3, AND WebGL , Second Edition, 2014, ISBN 978-1-118-85545-4

Patrick Carey New Perspectives on HTML5 and CSS3, 7th Edition, Comprehensive, 2016, ISBN: 978-1-305-50393-9

Rex van der Spuy Foundation Game Design with HTML5 and JavaScript, 2012, ISBN-13 (electronic): 978-1-4302-4717-3

Expected date of thesis defence

2017/18 SS – FEM

The Bachelor Thesis Supervisor

Ing. Martin Havránek, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 9. 1. 2018

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 12. 1. 2018

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 04. 03. 2018

Declaration

I declare that I have worked on my bachelor thesis titled "Game Development Using HTML5" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 14.03.2018

Viktoriia Pukha

Acknowledgement

I would like to thank Ing. Martin Havranek, Ph.D. for his advices, consultations and support during my work on this thesis. In addition, I would like to thank my family for reviewing my work, and support me.

Game Development Using HTML5

Abstract

The goal of this work is to introduce the process of a 2D web game creation using such technologies as HTML5, JavaScript, Phaser framework. The subgoal is to implement a platform game. The work consists of a brief description of HTML5, its main drawing API. Existing technologies which can be helpful during development are analysed within the comparison of most popular frameworks. The practical part dedicated to game implementation. It starts with working environment, assets, resources preparations. The outcome of the practical part is a platformer game, implemented with the help of Phaser framework.

Keywords: HTML5, JavaScript, Phaser, canvas, web, game, 2D

Tvorba her s využitím HTML5

Abstrakt

Cílem této práce je představit proces vytváření 2D webových her pomocí technologií HTML5, JavaScript a Phaser frameworku. Sekundárním cílem je naimplementovat platformovou hru. Práce se skládá ze stručného popisu HTML5, jeho hlavnímu API výkreslování. Stávající technologie, které mohou být užitečné během vývoje, jsou analyzovány v rámci porovnání nejpůlárnějších frameworku. Praktická část je věnovaná implementací her. Začíná uvodem do pracovního prostředí, použitými prostředky, přípravou zdrojů. Výsledkem praktické části je platformova hra, realizovaná s pomocí Phaser frameworku.

Klíčová slova: HTML5, JavaScript, Phaser, canvas, web, hra, 2D

Table of content

1	Introduction	10
2	Objectives and Methodology	11
2.1	Objectives.....	11
2.2	Methodology	11
2.3	Similar Solutions.....	11
3	Literature Review.....	13
3.1	HTML5.....	13
3.1.1	Drawing API.....	14
3.1.1.1	The Document Object Model and Canvas.....	14
3.1.1.2	WebGL	15
3.1.1.3	SVG	16
3.2	Game engines analysis	16
3.2.1	Construct2 – visual game maker.....	18
3.2.2	JavaScript game engines.....	20
3.2.3	PhaserJS	24
3.3	Browser debugging tools.....	25
3.3.1	Chrome Developer Tools.....	25
4	Practical Part.....	29
4.1	Development Environment Setup	29
4.1.1	Browser	29
4.1.2	Web server	29
4.1.3	Web development IDE.....	31
4.1.4	Other tools.....	31
4.2	Game description.....	31
4.3	Layout Design	32
4.4	Assets preparation	32
4.4.1	Tilemap drawing.....	32
4.4.2	Sprites	40
4.5	Implementation.....	41
4.5.1	Files structure.....	41
4.5.2	Physics	42
4.5.3	Player character.....	43
4.5.4	Debug.....	44
5	Results and Discussion.....	46
5.1	Future development.....	46

6 Conclusion	47
7 References	48

List of pictures

Figure 1. Drawing a line using canvas.....	15
Figure 2. Drawing a line using SVG.....	16
Figure 3. HTML and JavaScript game engine rating.....	17
Figure 4. Event system.....	20
Figure 5. Phaser states.....	24
Figure 6. Google.com in developer tools view.....	25
Figure 7. Chrome Network.....	26
Figure 8. Chrome console.....	26
Figure 9. Chrome extension.....	27
Figure 10. Safari Developer Tools.....	27
Figure 11. Browser & Platform Market Share.....	29
Figure 12. Running web server.....	30
Figure 13. Game Layout.....	32
Figure 14. New map creation with Tiled.....	33
Figure 15. Tiled map drawing.....	33
Figure 16. CSV file of a map.....	33
Figure 17. Map creation environment.....	34
Figure 18. Layer's options panel.....	34
Figure 19. Tiles.....	34
Figure 20. New tileset adding.....	35
Figure 21. Tileset added.....	35
Figure 22. One tile is selected.....	36
Figure 23. Group of tiles are selected.....	36
Figure 24. Background Layer.....	37
Figure 25. Obstacles level.....	37
Figure 26. Object Layer.....	38
Figure 27. Result of drawing.....	39
Figure 28. Player sprite sheet.....	40
Figure 29. Enemies sprite sheet.....	41
Figure 30. Project folder.....	41
Figure 31. Debug mode on.....	45

List of tables

Table 1. Licenses comparison.....	19
Table 2. Frameworks comparison (22.12.2017).....	23

1 Introduction

During last few years HTML5 and JavaScript actively getting a popularity among the game developers. The reason of that is very simple – the potential of HTML and JavaScript in game production. HTML5 is considered to be a powerful game platform. HTML5 games use the technology that allows them to run immediately, without different extensions installation. It is not required Flash¹, and it means that HTML5 enables a cross-platform functionality.

Using just HTML5, CSS², and JavaScript³ it is possible to draw lines, figures on the screen, to handle animation, to respond on user action. That technologies make possible to run video and to produce audio content. It supports forms that validate input and provide a feedback to users. HTML5 shows a great graphics, 2D as well as 3D.

Browsers and JavaScript are constantly getting more powerful and fully featured. There was a time when building any type of game required Flash. But with this out of the way, the stage is set for powerful HTML5.

¹ Adobe Flash is a multimedia software platform that used for production of desktop and mobile applications, games, video, animations. It displays graphics, text, allows streaming audio and video. Flash can capture mouse, keyboard, camera, microphone input.

² Cascading Style Sheets – is a stylesheet language, describes how HTML elements to be displayed on screen.

³ Scripting language, run in web browser to create dynamic content.

2 Objectives and Methodology

In this chapter, the objectives and methodologies are introduced.

2.1 Objectives

The work is dedicated to development of 2D game using HTML5. The main goal is to introduce the process of web game development with HTML5. The approach I used in this thesis is to explain HTML5, JavaScript concepts in the context of game development. The subgoal is to design and implementation of a platform game with selected technologies.

The main technologies for the web game development using HTML5 will be introduced. The task is to build a game, in a way to provide understanding of how we can use HTML5, JavaScript to develop a game.

2.2 Methodology

The work consists of brief analysis of technologies that can be used during 2D game development. Comparison of the main game engines is introduced. Implementation part starts from the setup of the working environment, mainly a web server, asserts preparation. At the beginning of that part I introduce software, that is needed, and where to find and download it. Implementation part describes a process of web game development, using technologies that were described in theoretical part. Phaser framework is used to create a platformer game.

2.3 Similar Solutions

There are few bachelors' works that have a deal with a development in HTML5. "*Use of technologies HTML5 and CSS3*", by Ruben Claudio Guarachi Torres, the work is dedicated to HTML5 and CSS3 for presenting a web content. The author focuses on functionalities of the technologies for a web site developing, mainly on responsiveness, validation process. He introduces canvas element, audio and video content. Another bachelor solution is "*Programming HTML5 games using the MelonJS framework*", by David Salcer. The author explains the process of game creation, using of MelounJS. The work is mainly dedicated to MelounJS framework. "*HTML5 hra*", written by Petr

Nemecek, who's goal was to implement an action arcade game. The work is focused only on the development process, using just HTML5 Canvas technology.

3 Literature Review

This chapter focused on overview of technologies that needed for a game development. The main game engines visual and scripting described, their comparison are introduced.

3.1 HTML5

HTML5 (HyperText Markup Language 5) – the language for structuring and presentation of the contents on the World Wide Web. HTML5, the latest version of the HTML standard, provides us with many new features for improved interactivity and media support. These new features (such as canvas, audio, and video) have made it possible to make fairly rich and interactive applications for the browser without requiring third-party plug-ins such as Flash. (1) HTML5 was completed only in 2014, the previous fourth version was published in 1999. From year 2013 web browser started to support the standard of HTML5. The purpose of HTML5 development was to improve the level of multimedia technology support. In the World Wide Web for a long time and to this day are used standards: HTML4, XHTML⁴ and XHTML1.1. HTML5 was created as a single markup language that could combine syntactic norms of HTML and XHTML. HTML5 improves, expands and makes rational markup documents. It adds a single API⁵ for complex web applications.

In HTML5 there many new elements that allow a programmer to reduce a number of lines of code. It allows to avoid of writing some elements using JavaScript, as a result its increases the total number of devices that view all the content presented on the page. For example, the elements <video>, <audio> and <canvas>, as well as the possibility of using SVG⁶ and mathematical formulas, now requires a few lines of code, instead of 10 as it was before HTML5. These innovations are designed to simplify the creation and management of graphics and multimedia objects in the network without the need of using third-party APIs and plug-ins. Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the semantic content of the document. Newest attributes were introduced for the same purpose, although a number of elements and attributes were removed. Some elements, such as <a>, <menu> and <cite>, were changed, redefined or

⁴ The Extensible HyperText Markup Language -

⁵ Application Programming Interface

⁶ Scalable Vector Graphics

standardized. API and DOM⁷ have become the main parts of the HTML5 specification. HTML5 also defines some features for error handling, so syntactic errors must be viewed the same by all browsers, that are compatible with HTML5.

The structure of an HTML5 file is very similar to that of files in previous versions of HTML except that it has a much simpler DOCTYPE tag at the beginning of the file. (2) Listing below provides a basic skeleton for a HTML5 file. Executing this code involves saving it as an HTML file and then opening the file in a web browser.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>The HTML5</title>
  <meta name="description" content="The HTML5">
  <meta name="author" content="Viktoriia Pukha">
  <link rel="stylesheet" href="css/styles.css?v=1.0">
  <!--[if lt IE 9]>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv.js
"></script>
  <![endif]-->
</head>
<body>
  <script src="js/scripts.js"></script>
</body>
</html>
```

3.1.1 Drawing API

One of the main feature of the HTML5 specification is the new drawing APIs: Canvas, SVG, WebGL. They provide bitmapped, vector, 3D drawing capabilities respectively.

3.1.1.1 The Document Object Model and Canvas

The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is

⁷ Document Object Model – represents all the objects on an HTML page.

accessed and manipulated. (3) The DOM allows the content to be updated after it is rendered in the web browser. The DOM is accessible through JavaScript.

The most important element in HTML5 game programming is the new canvas element.

The `<canvas>` element itself has no drawing abilities, using a script actually draw the graphics. The canvas element itself is accessible through the DOM in a web browser via the Canvas 2D context, but the individual graphical elements created on Canvas are not accessible to the DOM. (4) Canvas works in immediate mode, it does not have its own objects, it has only instructions on what to draw on a frame.

There are two DOM objects for a work with canvas: the window – that is the top level of the DOM and the document – that contains all HTML tags that are on the HTML page.

The simple canvas element that represents a line is written below.

```
<canvas id="mycanvas" width="200" height="100"></canvas>
```

To find canvas element in the DOM, `getElementById()` is used. The canvas element has a DOM method called `getContext`, used to obtain the rendering context and its drawing functions. This function takes one parameter, the type of context2D.

```
var canvas = document.getElementById("mycanvas");  
var ctx = c.getContext("2D");  
ctx.moveTo(0,0);  
ctx.lineTo(200,100);  
ctx.stroke();
```

The result:

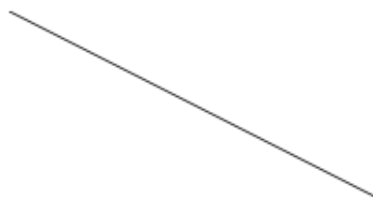


Figure 1. Drawing a line using canvas. Source: own processing

3.1.1.2 WebGL

WebGL (Web Graphics Library) is a JavaScript API for 3D drawing that enables the developer to assess graphics hardware and control minute details of the rendering pipeline.

(5) It is run in browsers through a `<canvas>` element after getting a context from defined

element. All calls are done through JavaScript. WebGL is widely supported in modern browsers.

3.1.1.3 SVG

SVG (Scalable Vector Graphics) is a mature W3C specification for drawing static or animated graphics. (5) HTML5 allows embedding SVG using `<svg> </svg>`, the ability to inline SVG without use of an object. The basic syntax is:

```
<svg xmlns="http://www.w3.org/2000/svg"> </svg>
```

Vector graphics use groupings of mathematics formulas to draw primitives such as arcs, lines, paths, and rectangles to create graphics that contain the same quality when rendered at any scale. This is a marked benefit over images whose discernible quality degrades when they are displayed at a scale larger than that for which they were designed.

To draw a line:

```
<svg id="svgline" height="200" xmlns="http://www.w3.org/2000/svg">  
<line x1="0" y1="0" x2="200" y2="100" style="stroke:rgb(10, 10, 10);stroke-  
width:2"/>
```

The result:

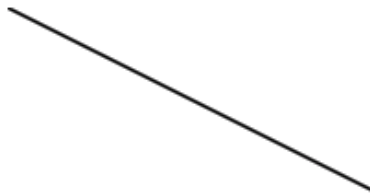


Figure 2. Drawing a line using SVG. Source: own processing

The difference between SVG and canvas approach of drawing is that SVG represents drawing in XML files instead of drawing in code. “XML is not the more concise representation of data, so a file may contain many repeated sections. This can be addressed by compressing the file, which can greatly reduce its size. As with the canvas element, interaction can be scripted using JavaScript. (5)”

3.2 Game engines analysis

Game engines could be arranged in two main groups: a visual game maker, and a scripting one. Most of developers are probably going to lean toward using a game engine or building from scratch, but there is also the alternative of using a visual game maker like Construct2.

Using a game maker means not actually write in JavaScript, instead, to create code-like events in the editor. It's a trade of ease-of-use and quickness to prototype or develop vs customization and control over the end result. There are plenty JavaScript frameworks as well as HTML engines that make a game development process much easier and interactive. But without analysis of each of it, it is impossible to choose the right one. Examples of visual game engines are Construct2, GameMake. Scripting frameworks: phaser, MelounJS, pixi.js, Three.js, Panda.js, etc.

Every engine has something different to offer to developers. HTML5GameEngine.com is a web site that provide the long list of HTML5 and JavaScript game engines from the most popular to the less. Each engine is tied to important information about it: popularity, user ratings, latest release, and details, that provide users review.

The popularity and sample game information is grabbed from a spider that crawls the web in search of HTML5 games. Once it grabs games, it dissects them and categorizes by the game engine used to build them. (6)

Name	Cost	Popularity	Rating	Tags	Last Release	Details
Construct 2	varies		★★★★☆	game-maker free 2d 3d webgl sounds collisions physics	Aug 19th 2014	More Details
ImpactJS	\$99		★★★★☆	2d sounds collisions physics debug map-editor	Jul 28th 2014	More Details
EaselJS	free (MIT)		★★★★☆	flash-like 2d sounds free	Jun 1st 2017	More Details
Phaser	free (MIT)		★★★★☆	flash-like 2d sounds collisions physics typescript webgl free	Jun 2nd 2017	More Details
pixi.js	free (MIT)		★★★★☆	2d webgl free	May 18th 2017	More Details
GameMaker	\$200		★★★★☆	game-maker 2d sounds collisions physics debug map-editor	Aug 8th 2014	More Details
Three.js	free (MIT)		★★★★☆	3d webgl free	May 30th 2017	More Details
PlayCanvas	free		★★★★☆	3d cloud-based free webgl sounds	Jun 2nd 2017	More Details
Turbulenz	free (MIT)		★★★★☆	2d 3d webgl sounds collisions physics debug networking	Dec 22nd 2015	More Details
lycheeJS	free (MIT)		★★★★☆	2d sounds debug ui networking	Mar 27th 2017	More Details
melonJS	free (MIT)		★★★★☆	2d sounds collisions physics free map-editor	May 3rd 2017	More Details
Cocos2d-X	free (MIT)		★★★★☆	ios-like free 2d physics	Apr 30th 2015	More Details
Quintus	free (MIT)		★★★★☆	jquery-like 2d sounds free	Feb 2nd 2016	More Details
WADE	free (varies)		★★★★☆	2d isometric modular physics	Aug 5th 2014	More Details
Crafty	free (MIT)		★★★★☆	free 2d sounds collisions	Feb 12th 2017	More Details
enchant.js	free (MIT)		★★★★☆	2d sounds collisions physics webgl free	Jan 4th 2016	More Details
LimeJS	free (Apache)		★★★★☆	2d sounds physics free	Jun 1st 2015	More Details
Isogenic Engine	varies		★★★★☆	2d isometric physics path-finding networking	Feb 24th 2014	More Details
Panda.js	free (MIT)		★★★★☆	free 2d webgl mobile physics sounds modular	Feb 17th 2015	More Details
Kiwi.js	free (MIT)		★★★★☆	2d webgl physics free	Nov 15th 2015	More Details
GC DevKit	free (Mozilla)		★★★★☆	2d mobile-first sounds collisions physics debug	Mar 24th 2016	More Details
voxel.js	free (BSD)		★★★★☆	webgl 3d voxel sounds physics networking	Oct 4th 2015	More Details

Figure 3. HTML and JavaScript game engine rating. Source: (6)

3.2.1 Construct2 – visual game maker

Construct2 is an HTML5-based game editor with a lot of features, enough for people beginning to work with game development to make their first 2D game. (7)

It designed specifically for 2D games. Construct2 has a drag and drop nature, it is not required programming background.

The main features of this engine are:

- Multiplatform development. Games can be published to desktop computers, to many mobile devices, and also to a web site using HTML5, even to publish it on Nintendo's Wii U.
- Easy use for non-programmers. It relies on event system, without coding experience.
- Built-in physics. It provides an approximate simulation of physical system, like collision detection, or fluid dynamics, etc.
- Instant preview allows or preview a game at any time, no need to wait for compiling.
- Interactive development which enables testing during creation process.
- Can be extended. Many plugins are available from third-party developers.

Construct2 is covered by proprietary license⁸. There are 3 license plans are available on the official web site. A trial version is also available.

	Personal Licenses	Business Organizations	Education Organizations
Price per year	2 271,49 CZK	From 3 306,69 CZK	919,99 CZK

⁸ Proprietary software is computer software that has restriction on copying and using it. A publisher retains intellectual property rights.

Who can buy	For individuals and hobbyists who want to create games.	For business of all sizes and types. But the price varies depending on business size. If the revenue of the company is more than \$50 000 the price is 8 886,39 CZK	For all types of educational institutes and organizations. But not for students. Only teachers and administrators can purchase the plan
Commercial use	Limited. The revenue from the game should not exceed \$5000	Unlimited.	Not permitted.

Table 1. Licenses comparison. Source: (8)

Construct2 is that is a desktop app, so it need to be installed and it has systems requirements are provided below:

Minimum system requirements

Windows XP Service Pack 3 or newer

512 MB RAM

1 GHz Processor

A HTML5 compatible browser

The latest version of your graphics card drivers (9)

Construct2 provides visual game creation. It is possible even to draw characters in the software and test it, so user can immediately decide is the character is suitable for a game or not. Construct2 runs everything in the event sheet once per tick⁹. Events consist of conditions, depends on certain criteria the event's actions are run.

⁹ Most monitors update their display 60 times per second, so Construct2 event sheet usually run 60 times per second.


12	Bullet	On collision with Enemy	Enemy	Subtract 1 from Health
			Bullet	Destroy
			System	Create object  redDamage on layer "Game" at (Bullet.X, Bullet.Y)
			Add action	
13	Enemy	Health ≤ 0	Enemy	Destroy
			System	Add 200 to score
			Add action	

Figure 4. Event system. Source: (7)

Figure 4 shows, when any bullet collides with any enemy, the actions on its right-hand side will be executed. In this case, it will subtract the enemy's health, destroy the bullet object, and create a new object for a damage effect. The next event, number 13, is what happens when an enemy's health drops below zero; the actions will destroy the enemy and add points to the score variable. (7)

Summary. The advantages and disadvantages of the Construct2, depend on the needs and expectations of a user. For those who has no programming experience this engine is a good choice. Disadvantage of that could be if users want to learn how to code during development, in that case this engine is not for them. Another disadvantage is a price, the free edition is very limited in used. Mainly debugger tab is not available as well as event breakpoints, it is not permitted to make Windows, Mac, Linux apps, limited quantity of sound effects. There are plenty examples and good documentation in form of manual and tutorials is available on the official web site¹⁰.

3.2.2 JavaScript game engines

The popularity of JavaScript has led to a creation of many frameworks. Along with all of the new technologies that were created to make our life easier, a higher degree of confusion for many people appeared in choosing the best one. The one that will suite an application the best.

¹⁰ <https://www.scirra.com/manual/1/construct-2>

To choose frameworks that can be used for game development, the comparison method was used. The following criterias was selected, based on the initial state of the game and it future development plans.

The criterias are:

- Well documented. Tutorials, examples, manuals;
- Developer Support.
- 3rd party integration
- Maintained. Last release, the frequencies of updates;

There are 6 most popular JavaScript frameworks were selected for comparison. The information was taken from the official web sites, and forums, communities, as well as from own processing. The results are summarized in the table below.

	Documentation	Support	Integration	Last Release
EaselJS	Well documented. Five official tutorials. Seven community tutorials. Few examples and articles.	No developer supports.	TweenJS for a work with tweening and animation. SoundJS for a work with audio. PreloadJS for a managing and coordination of the loading assets and data. Supports tools like Zoe and Adobe Animate.	18.09.2017
Phaser	Well documented. Online tutorials and examples on the official web site.	Support services includes: <ul style="list-style-type: none"> • Pre-project planning. • Checking Phaser is suitable for your specific 	<ul style="list-style-type: none"> • Tiled • TexturePacker • ShoeBox • Audiosprite • http-server • uglify-js • Phaser generator 	21.12.2017

		<p>requirements.</p> <ul style="list-style-type: none"> • Help writing technical specifications and technical checking proposals. • Talking with development team about the best practices. • Help fixing gnarly bugs. • Helping with on-device testing. • Creating custom builds. • Integration with 3rd party systems. 	<ul style="list-style-type: none"> • CocoonJS 	
PixiJS	Great documentation, tutorials, dev forum, FAQ and blow. Three published books.	Game Dev forum.	Rendering engine that uses WebGL with canvas fallback. Allows WebGL filters and provide a scene graph. It is not a full game engine, it is just rendering engine.	30.11.2017
LycheeJS	Live demos. Guide contains tutorials, troubleshooting	No developer supports.	Peer-cloud, the software automation pipeline and all running software bots.	20.12.2017

	help, concept explanations, hints and tricks. Poor number of examples.		Project management wizard, graphical editor, build tool for packaging, testing tool.	
MelonJS	Blog, tutorials, forum.	No developer support.	<ul style="list-style-type: none"> • Tiled • TexturePacker • Shoebox • PhysicEditor • BitmapFont Generator • Cordova/PhoneGag • CocoonJS • Ejecta 	3.12.2017
ThreeJS	Forum, plenty examples, Slack	No developer support	<p>Has its own visual editor. Supports physically based rendering, allows real life quality material and lighting.</p> <p>Poor garbage collector, causing issue with the performance of a project.</p>	18.12.2017

Table 2. Frameworks comparison (22.12.2017)

For the Practical part Phaser.js framework was selected. It is very well documented, there are a lot of examples on the official web site, API documentation, tutorials. Phaser has a large community, from chat rooms, forums to weekly newsletters.

Developers of Phaser offer premium technical support services to companies. All support is private and confidential. Support is charged on an hourly basis and can be purchase online.

There are a lot of 3rd party tools that can be integrated to Phaser projects:

1. TexturePacker to pack individual sprites into one single spritesheet.
2. ShoeBox contains many tools, such as spritesheet and bitmap font generators.
3. Audiosprite to generate audio sprites.
4. Uglify.js to concatenate and minify JavaScript code.

Phaser is an open source project, it is actively developing and available on GitHub.

3.2.3 PhaserJS

Phaser is a free game framework that works via HTML and JavaScript. It offers Canvas and WebGL rendering across desktop and mobile web browsers. JavaScript or TypeScript can be used for development.

The core concept in Phaser is a state. A state is implemented using a JavaScript object and allows to split game code into logical chunks.

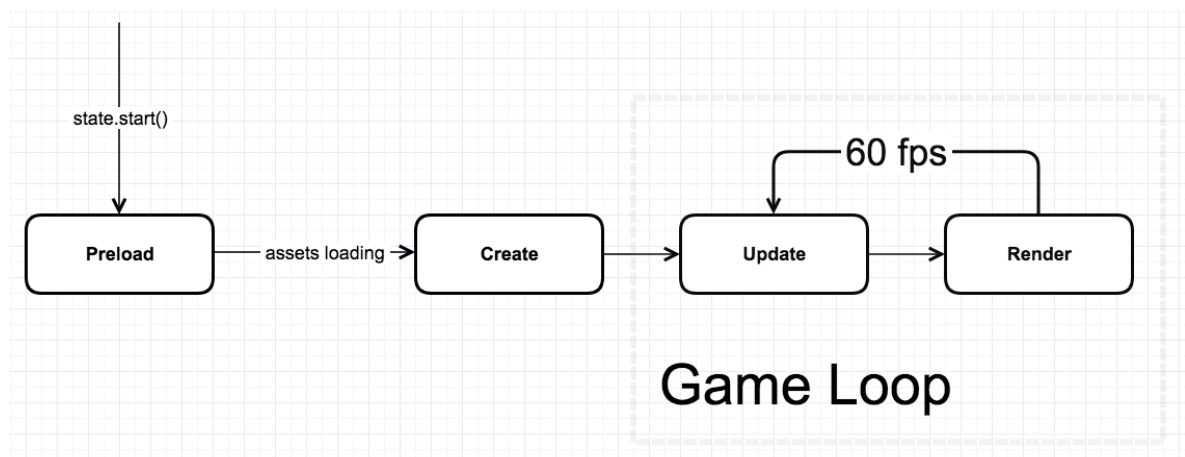


Figure 5. Phaser states. Source: own processing

- **Preload** – The game starts with this function. All resources are preloaded. We are using it to load a map, sprite sheets, audio, etc.
- **Create** - After preloading all assets, we can now setup the initial state of the game. We create objects from the map, assign assets, create layers.
- **Update** – Called to update the game state, at a set interval (usually 60 times per second). All updates to the game are done here. Checking collisions between player and layers.
- **Render** - Coming after Update, here is where the latest state of the game is drawn to the screen. For example, debug.

3.3 Browser debugging tools

The most important tool during web development is a browser, which implements most of the specifications and has many debugging tools. Google Chrome, Mozilla Firefox Apple Safari, and Opera have outstanding HTML5 compliance and debugging tools.

3.3.1 Chrome Developer Tools

Chrome Developer Tools allows to inspect the DOM dynamically, view resource loading times and run arbitrary JavaScript. On the picture below there is a console window for Google.com. It shows a nested view of the DOM with styles for document elements, when the Elements tab selected. Hovering over an element tag will highlight it in the browser window. This is really useful when trying to figure out which element is the one that is off by several pixels.

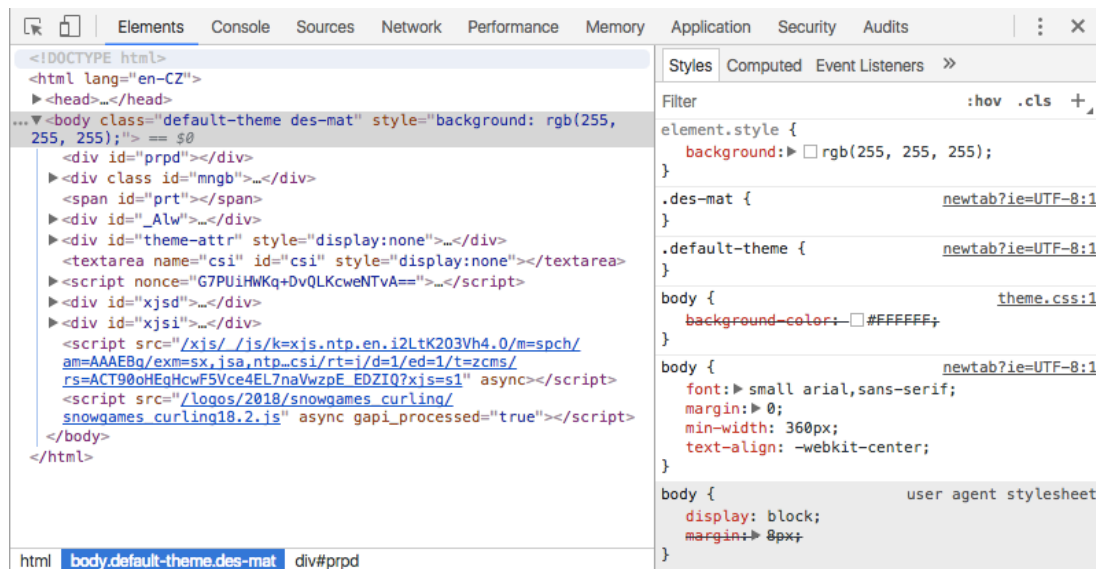


Figure 6. Google.com in developer tools view. Source: own processing

The other two tabs in the Developer Tools console that are incredibly useful for the game developers are the Network and JavaScript Console tabs. The Network tab, shown in Figure 7, allows to track on an asset-by-asset basis what exactly is slowing a web page from showing in Waves quickly. The first time you run it on a new site, you can decide to activate it just for this session or forever. Resource tracking does a bit of over- head to page loading times, so it is best to use it sparingly.

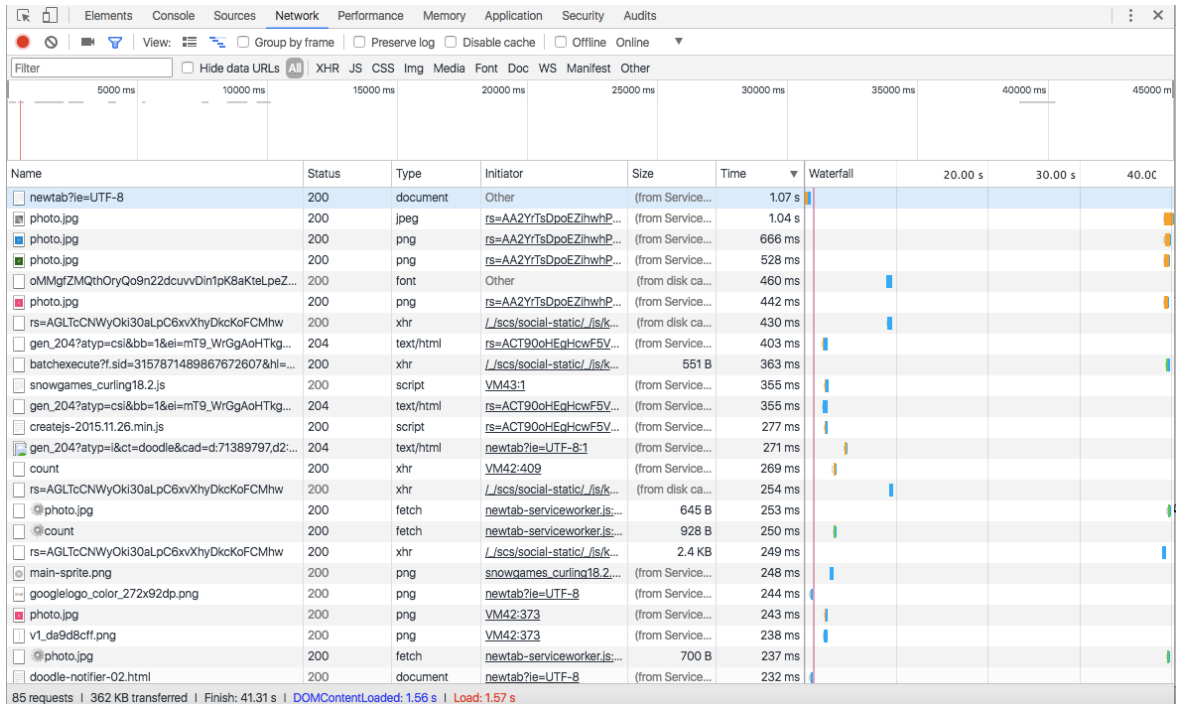


Figure 7. Chrome Network. Source: own processing

The last tab I'll highlight in this section is the JavaScript Console tab, as shown in Figure 8. It enables also use it to inspect the DOM programmatically or run arbitrary JavaScript.

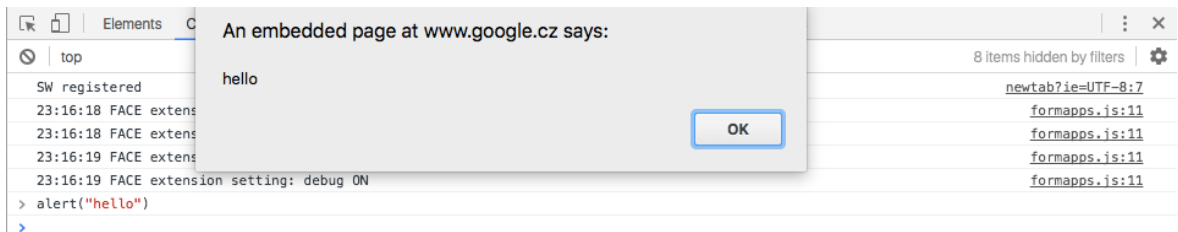


Figure 8. Chrome console. Source: own processing

Chrome Extensions

The functionality of Google Chrome can be enhanced and extended with extensions. A full list can be found at <https://chrome.google.com/extensions>. The installation is very easy, only by clicking the Install button from Chrome Extensions page, as shown on the picture below.

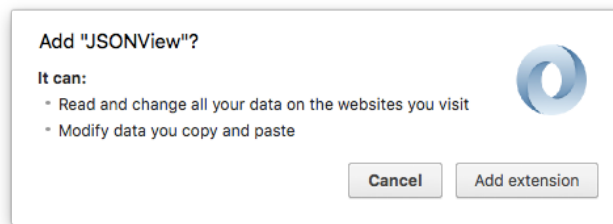


Figure 9. Chrome extension. Source: own processing

Two useful plugins for developers are JSONView and YSlow. JSONView allows you to view JSON data formatted to increase readability. YSlow analyzes web pages and gives tips on how to improve performance.

Safari Developer Tools

Apple’s Safari developer tools are very similar to Google Chrome. They are hidden to the end user, by default. They can be enabled by selecting Preferences in title bar and navigating to the Advanced tab. As shown in Figure 10, “Show Develop menu in menu bar” is checked.

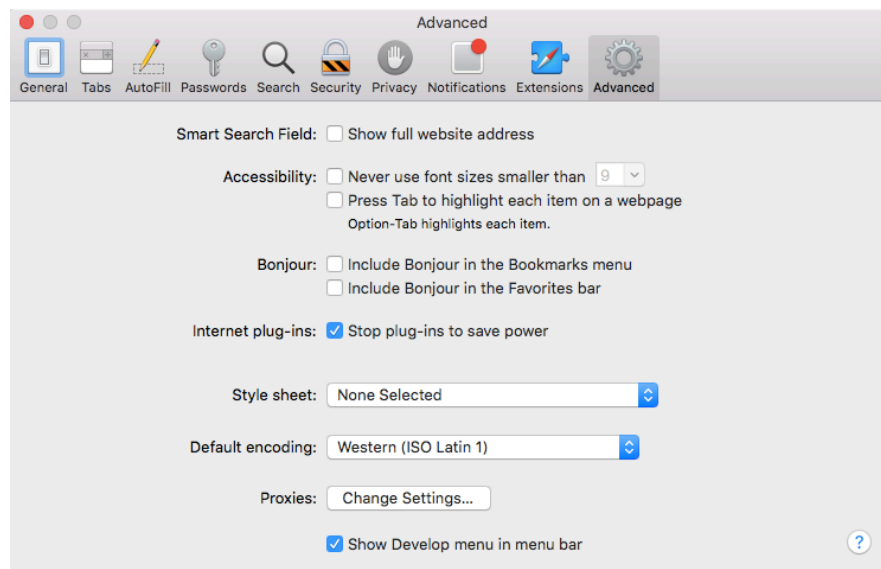


Figure 10. Safari Developer Tools. Source: own processing

Firebug

Firebug is an extension for Mozilla Firefox that allows developers to debug a website’s HTML, CSS, and JavaScript. Although originally designed for Firefox, Firebug also has a

“Lite” version that will run in Google Chrome and complements the tools already present in that browser. The core components and tabs are very similar to Chrome and Safari. As in Chrome, there are add-ons for Firefox and Firebug to expose more developer capabilities, such as DOM manipulation and introspection for several programming languages, such as PHP and Python.

4 Practical Part

In this chapter, the development process is described. Starting from an environment setup, and follows with a survey about a game genre selection, implementation and finishing with testing of the application. The practical part of the thesis provides a full how-to manual to implement a platform game.

4.1 Development Environment Setup

4.1.1 Browser

To make a web game first of all a browser that supports HTML5 (e.g. Chrome, Firefox) is needed. According to the Browser & Platform Market Share by November 2017. This report was generated **11/30/2017** based on the past month's traffic to all websites that use W3Counter's free web stat. (11)

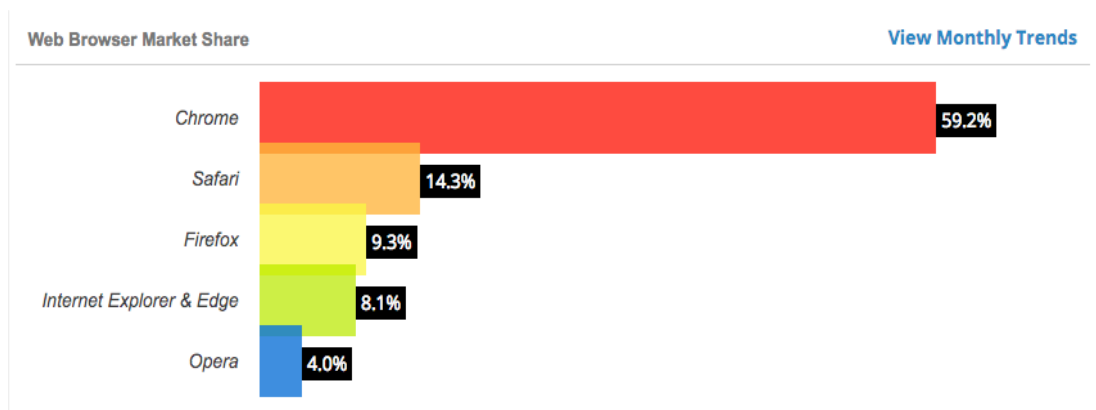


Figure 11. Browser & Platform Market Share. Source: (11)

To test any web application, strong debugging tools are required. Taking into consideration browser debugging tool, I choose Google Chrome version 63.0.3239.84.

4.1.2 Web server

To test an application, we need to use a web server. There are a lot of alternatives to choose a how-to setup a web server. I chose a simple NodeJS http-server. The web server runs on the http-server npm package. It is a simple zero-configuration http server for serving static files to the browser. The server is started from the command line and doesn't require a server.js file. First of all, NodeJS must to be downloaded from <https://nodejs.org>

and installed using all the default options. The next step I installed the http-server package from npm.

```
npm install -g http-server
```

After installation completed I started a web server from a directory containing my game static files.

```
cd Desktop/czuB/game
```

The next step is to start the server

```
http-server
```

After successful run the following message will appear:

```
Starting up http-server, serving ./
Available on:
  http://127.0.0.1:8080
  http://192.168.0.100:8080
Hit CTRL-C to stop the server
```

The game is started on localhost:8080

After successful running a server, to test it, open browser using the address where the web server is running.

In terminal, we will get the following:

```
[Sat Dec 30 2017 15:06:16 GMT+0100 (CET)] "GET /" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36"
```

After opening a link in the browser, the following window will appear. The server is running in empty folder, if there are some files, the file tree will be displayed.

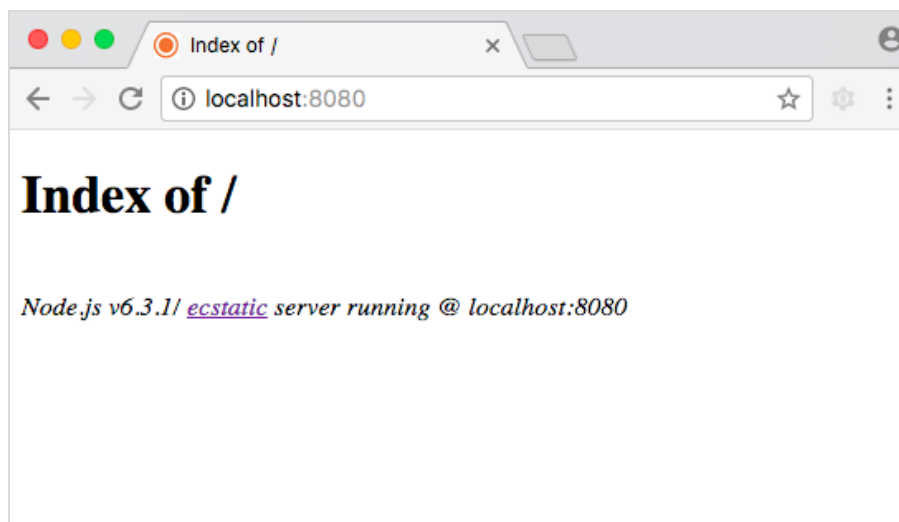


Figure 12. Running web server. Source: own processing

4.1.3 Web development IDE¹¹

It is needed a simple plaintext editor to edit HTML code, it depends on the skills and user creativity. For more comfortable work, especially for a work with JavaScript it is better to find some powerful IDE that suits all the user requirement. My choice is Visual Studio Code. It is a powerful code editor, that runs on desktop. It is available for Windows, Mac and Linux. It comes with built-in support for JavaScript. Visual Studio goes with syntax highlighting and autocomplete with IntelliSense. IntelliSense is a general term for a variety of code editing features including: code completion, parameter info, quick info, and member lists. IntelliSense features are sometimes called by other names such as "code completion", "content assist", and "code hinting." (12)

4.1.4 Other tools

Before game implementation, we will need the following tools:

- For a map creation Tiled map editor, is available from www.mapeditor.org
- Phaser CE available form <https://phaser.io/download/stable>
- Phaser Documentation, <https://phaser.io/docs/2.6.2/index>

4.2 Game description

One player plays a game. Player navigate its environment by jumping and running on platforms, while avoiding obstacles. The task is to collect as many coins as possible, avoiding and killing enemies. Player collect a coin by collide with it. Player kills enemy by jumping on it. If the player intersects with enemy, player will lose one life. If there is only the last one life, and player collides with enemy, the game is over. User can start to play the game again.

Game objects:

- A player – movements are controlled by arrow keys
- Obstacles – stones which player cannot go through
- Coins – that should be collected by player
- Enemies – that need to be killed by player

¹¹ An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. (2)

4.3 Layout Design

The game layout is very simple. There are 3 screens:

1. The Welcome screen. There is a logo, and a play button, that start a game. In future it will be extend with settings.
2. Loading bar. There is a waiting time, before all assets are loaded.
3. The game environment.

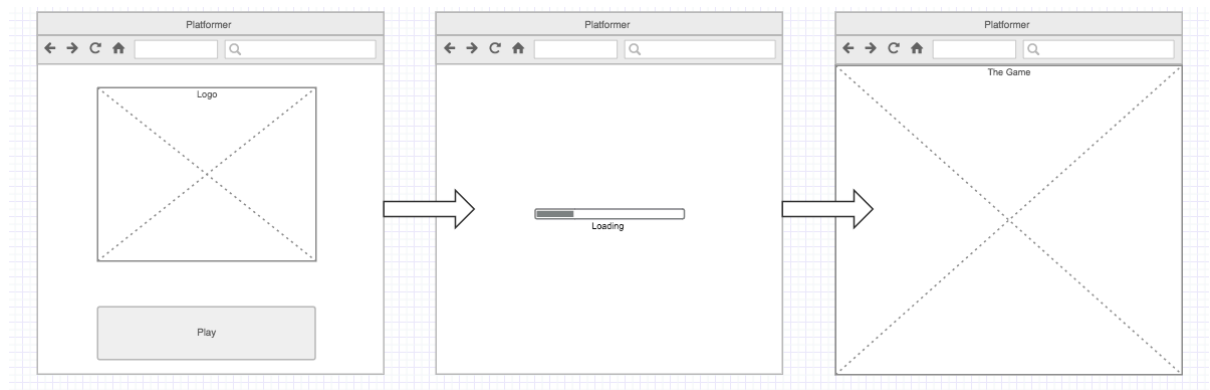


Figure 13. Game Layout. Source: own processing

4.4 Assets preparation

Game assets is everything that can go into a game, including images, sound effect, sprites, music.

The required assents are: a map, where the player will move, tiles, images, sprites, audio. All of the assets, used in game are free, can be used under creative common ¹²license.

4.4.1 Tilemap drawing

One of the advantages of using Phaser framework for game creation is the integration with a Tilemap. It allows to create levels using tiles and grid overlay. A tile is an image, which like a puzzle piece for building a larger image. Usually, tiles are rectangular or isometric. A map is consisting from tiles. It is more efficiency in checking of collisions, for example, it is no need to detect it on a pixel side but using a quick formula to find which tile to access. There few free resources on the internet, where tiles can be downloaded. I choose <https://opengameart.org>, there are plenty tiles and sprites can be found for a game.

¹² it is free to share – copy and redistribute the material in any medium or format, adapt – remix, transform, and build upon the material for any purpose, even commercially. (13)

For a creation of a map, I choose Tiled Map Editor, it is free, well documented, available for Windows, Linux and Mac OS X. The installation is very intuitive, I installed it with default options.

To create a new map, open Tiled and select File, New. A new window appeared, allows to input the following parameters: Orientation: Orthogonal, Map size: Width: 64 tiles (1024 px), Height: 64 tiles, Tile size: Width: 16px Height: 16px.

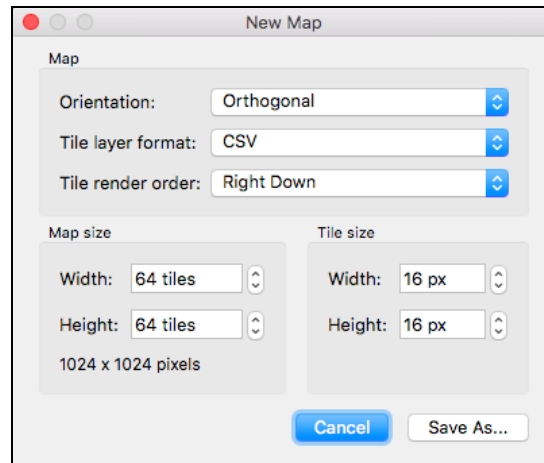


Figure 14. New map creation with Tiled. Source: own processing

The map can be saved in JSON, JS or TMX formats, but for using map in Phaser JSON or CSV format must to be chosen. After drawing the map will be like on the picture below:

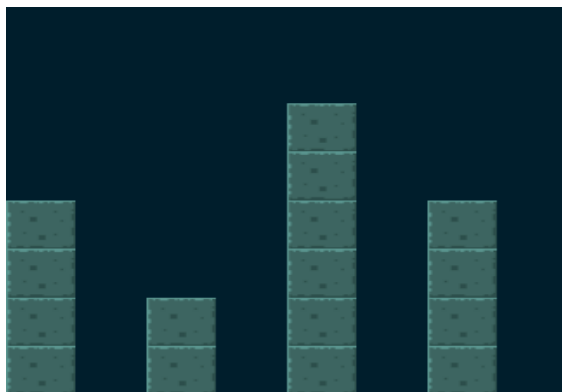


Figure 15. Tiled map drawing. Source: own processing

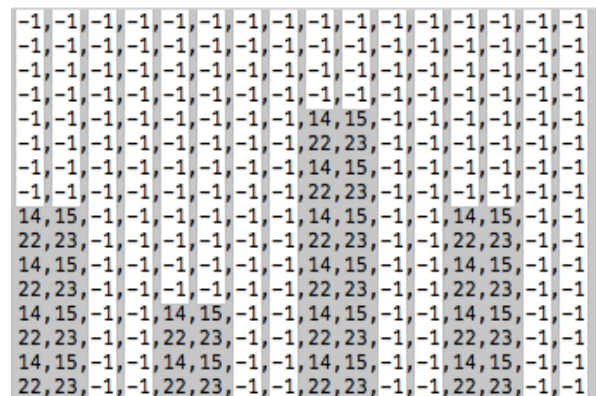


Figure 16. CSV file of a map. Source: own processing

I chose JSON format, I am more familiar with it. There is no much difference in proposed export formats. The only difference when using a CSV format, additional parameters during map loading in Phaser must be provided: tileWidth and tileHeight. After saving a map, new working layout will appear. On the top right part “The Layer 1” has already created.

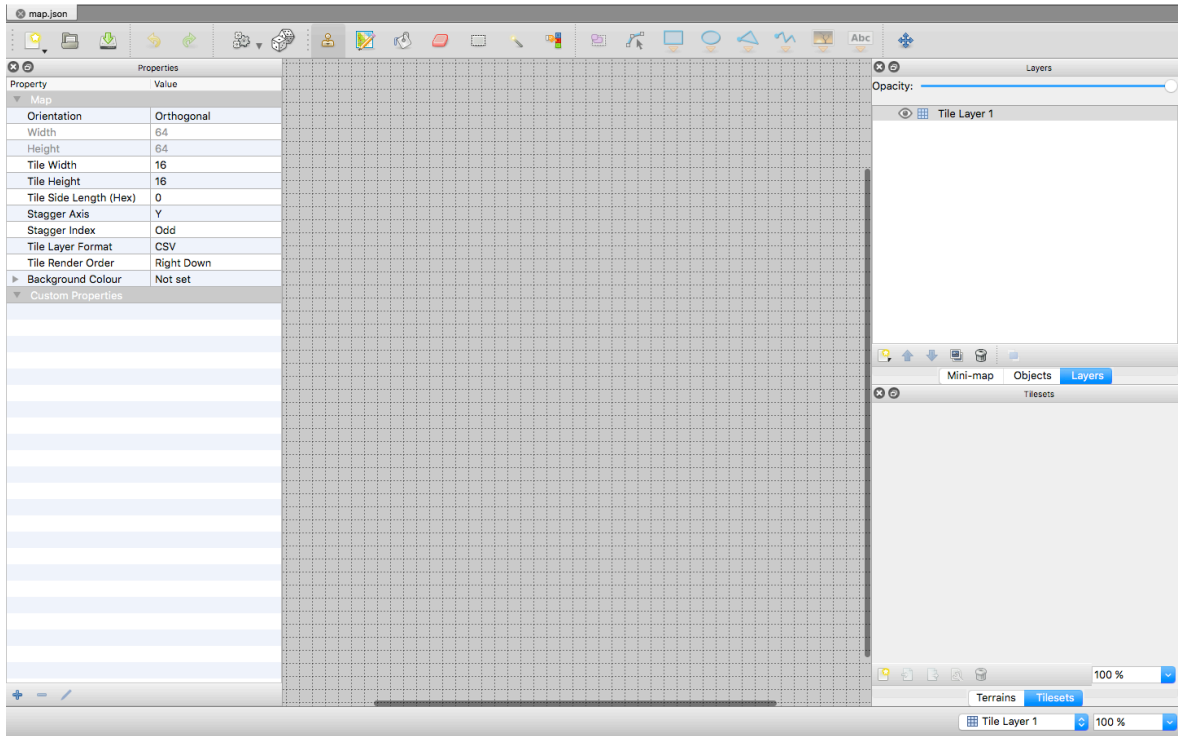


Figure 17. Map creation environment. Source: own processing

Below the layers area some options are available: create a new layer, raise layer, lower layer, duplicate, remove and show/hide all layers.



Figure 18. Layer's options panel. Source: own processing

The next step is to add some tiles on the map.

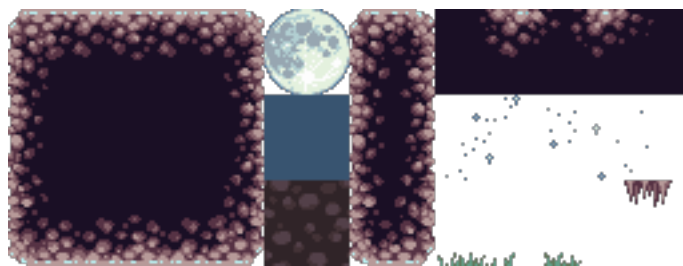


Figure 19. Tiles. Source: own processing

After creating a new tileset, the pop-up window as shown on the figure 20 will appear. Parameters that need to be set:

- Name - will be used during uploading a tile in Phaser
- Type: Based on Tileset Image – a set with a fixed size of tiles, allows to select margin and spacing between the tiles; Collection of Images – type of tileset, each tile refers to its own image, useful when the tiles size is not the same.
- Source of file – needed to be in the project assets folder.
- Embedded in map – tileset can be embedded in a map file or saved externally, possible to change later.
- Use transparent color – transparent background color of the tile.
- Width 16px, height: 16px – correspond to a real tile size
- Margin around and spacing between tiles.

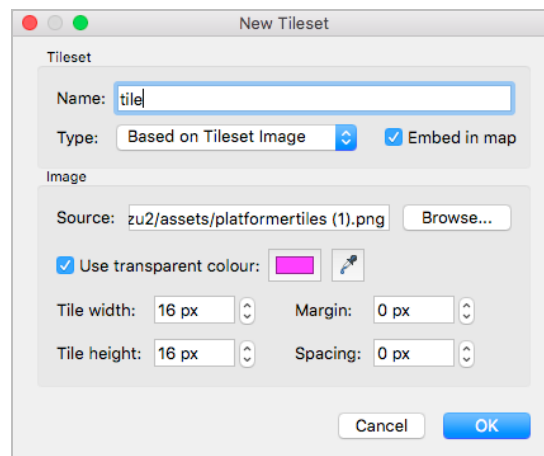


Figure 20. New tileset adding. Source: own processing

After adding a new tileset, it will appear in the left bottom part of the working area.

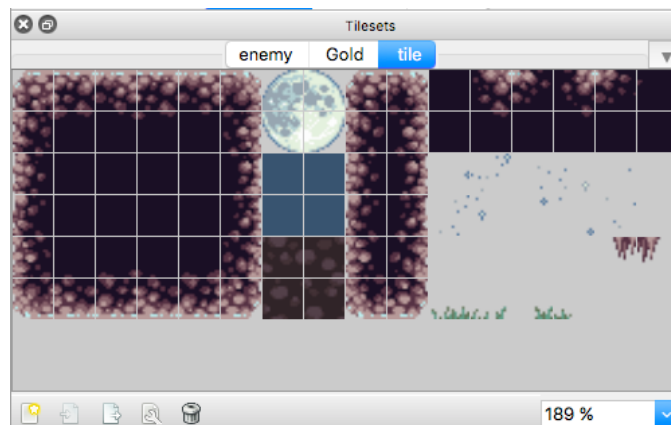


Figure 21. Tileset added. Source: own processing

The next step is the work with layers. There two main types of layers:

- Tiled Layer – used for everything that does not need custom properties. Background and collisions will be placed on a tile layer.
- Object Layer – stores many kinds of information, it has individual custom properties. This layer will be used in storing enemies and golds objects.

The lowest layer is a Background. To start drawing firstly the layer must be selected, secondly a tile, one or a group of them.

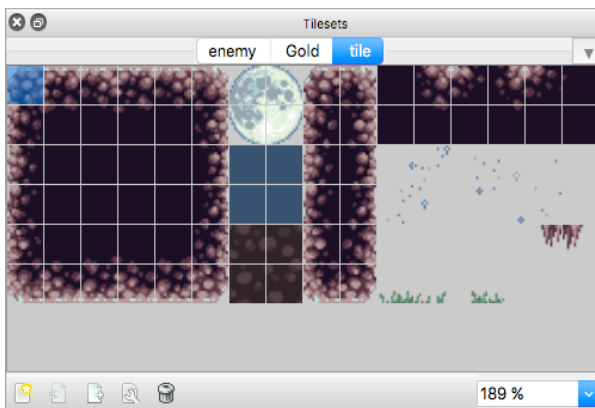


Figure 22. One tile is selected. Source: own processing

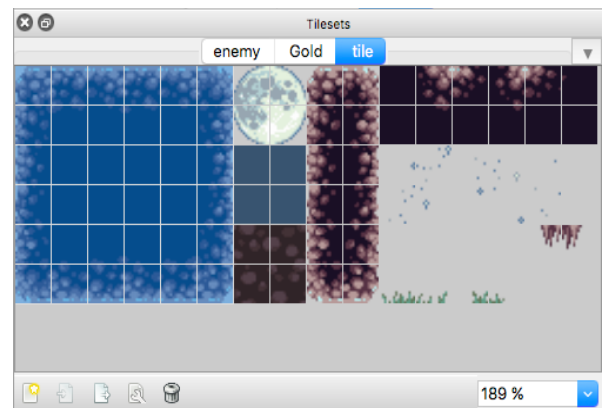


Figure 23. Group of tiles are selected. Source: own processing

Using stamp brush



we can start drawing on the map, using bucket fill



we can fill everything with a selected tiles.

My background layer is presented on the picture below:

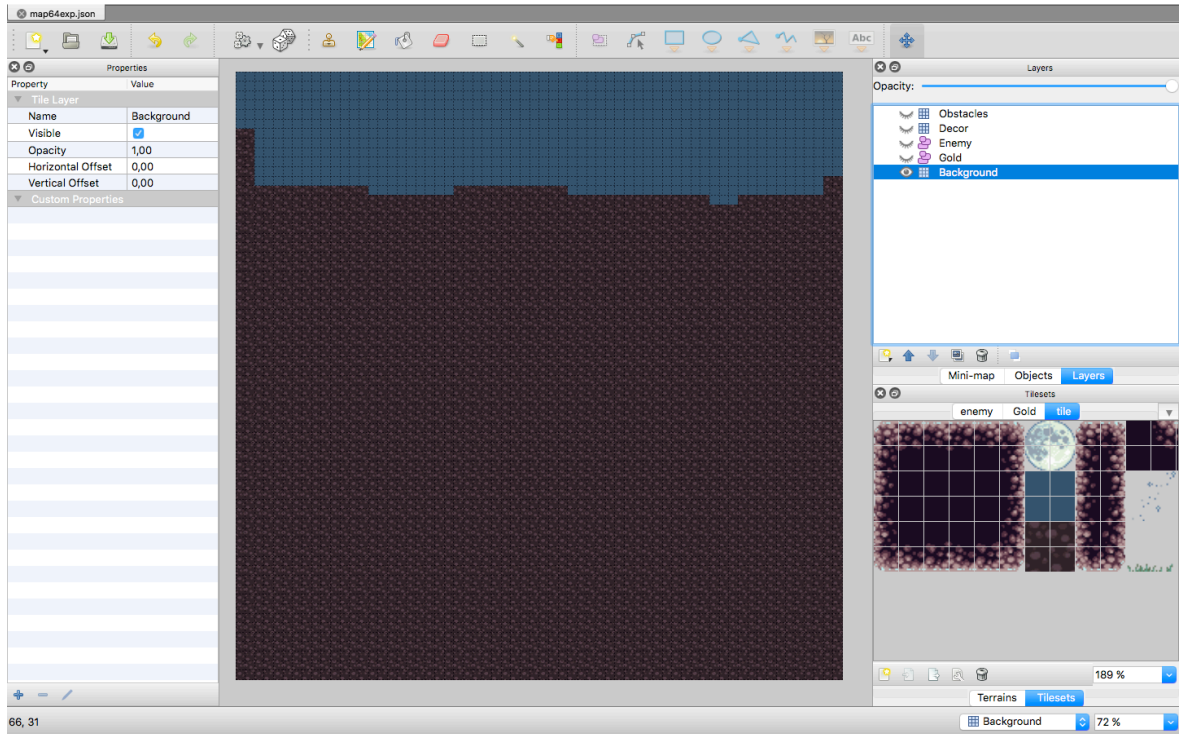


Figure 24. Background Layer. Source: own processing

The same principle is used to draw other tiled layers. Decor Layer – contains just some decorations on the map, like a grass, a moon, etc. Obstacle layer is a layer that will collide with a player. The layer represents walls that cannot be crossed.

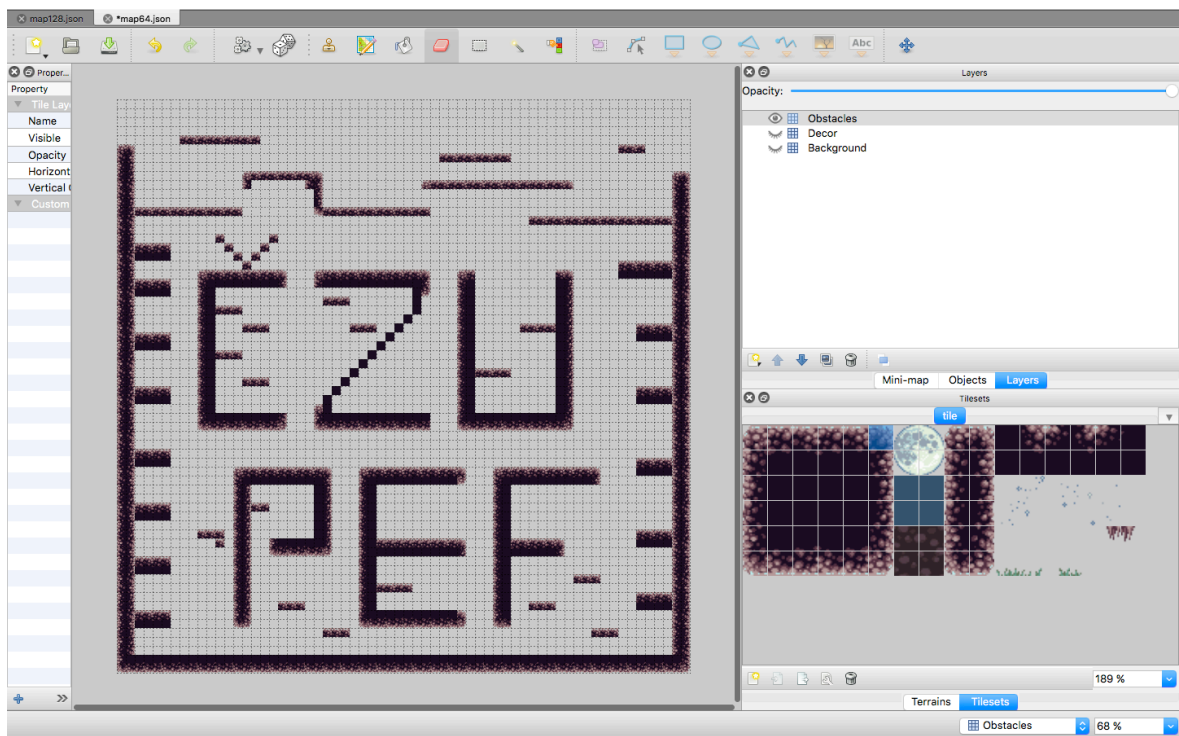


Figure 25. Obstacles level. Source: own processing

The next step is to add objects to the map. The objects are: golds and enemies. Player will collide with that objects, as a result some conditions will be met, and certain actions will be run.

Two layers need to be created: Enemy and Gold. I am selecting the Enemy layer on which enemies will be added. I am using “Rectangle Insert” button to add objects. Select Objects tab, as on the figure below, all objects that added on the Enemy layer are listed. For a later work with enemy objects, it is needed to give a name to all of them.

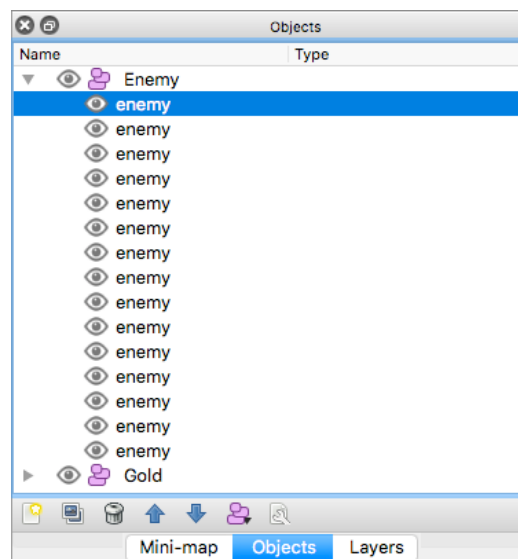


Figure 26. Object Layer. Source: own processing

The result of drawing of all layers are presented on the picture below:

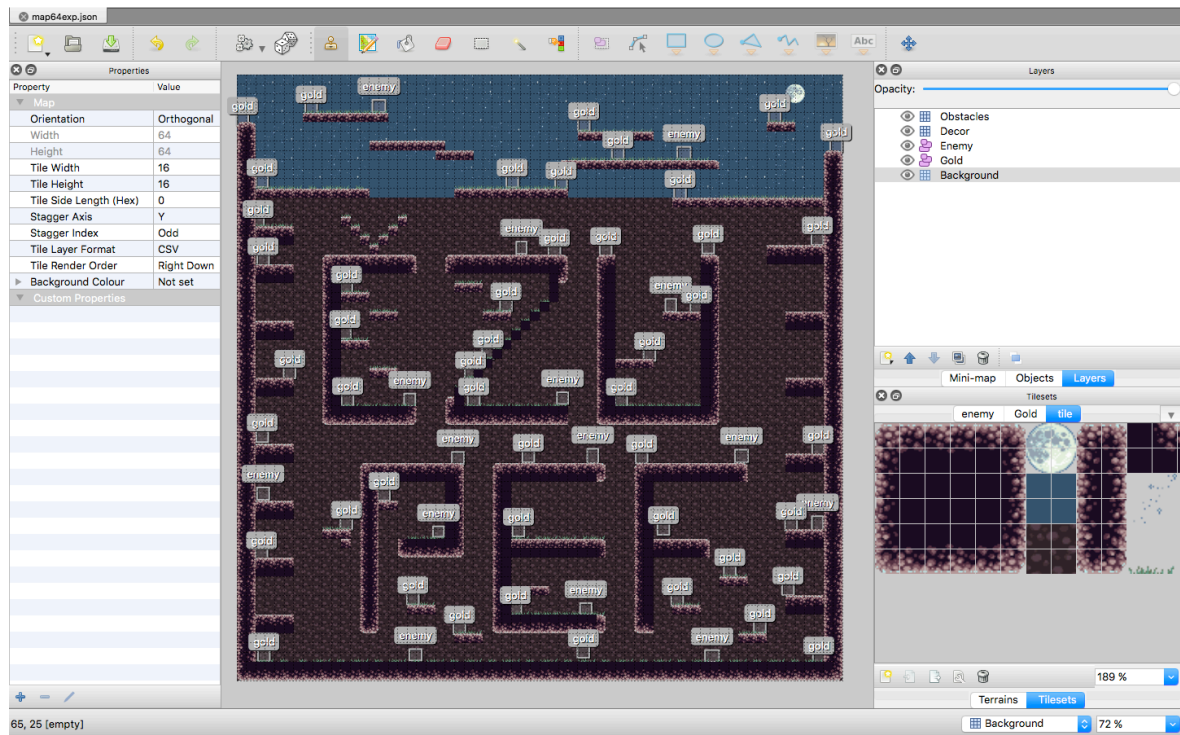


Figure 27. Result of drawing. Source: own processing

The tilemap is exported to JSON file. Each layer is represented in an array, where the numbers in “data” array are the IDs of each tile in a tileset. Numeration of IDs is started in the left upper corner from number 0.

```
"layers": [
  {
    "data": [39, 39, 39, 55, 55, 55, 55, 55, 55, 55, 55, 39, 40, 40, 40, 40,
40, 39, 55, 55, 55, 55, 55, 55, 55, 55, 39, 40, 40, 40, 56, 56, 39, 39, 39, 39, 39,
39, 39, 39, 39, 39, 39, 39, 39, 39, 40, 40, 40, 56, 56, 56, 40, 40, 40, 40, 39,
40, 40, 40, 40, 40, 40, 55, 56, 39, 55, 39, 55, 56, 39, 40, 39, 39, 39, 39, 39,
39, 39, 39, 40, 39, 55, 55, 56, 39, 40, 40, 40, 40, 40, 39, 40, 40, 40, 40, 55],
    "height": 64,
    "name": "Background",
    "opacity": 1,
    "type": "tilelayer",
    "visible": true,
    "width": 64,
    "x": 0,
    "y": 0
  }
]
```

The tileset is represented like:

```
"tilesets": [
  {
    "columns":16,
    "firstgid":1,
    "image":"platformertiles (1).png",
    "imageheight":96,
    "imagewidth":256,
    "margin":0,
    "name":"tile",
    "spacing":0,
    "tilecount":96,
    "tileheight":16,
    "tilewidth":16,
    "transparentcolor":"#ff00ff"
  }
]
```

4.4.2 Sprites

To animate characters in a HTML5, we need a sprite. Sprite is drawn on a canvas and animated through JavaScript. Animation can be achieved by rendering an area of a single image on a certain interval. A sprite sheet consists of many frames in one picture. The following sprite sheet, will be used for a player character in a game:

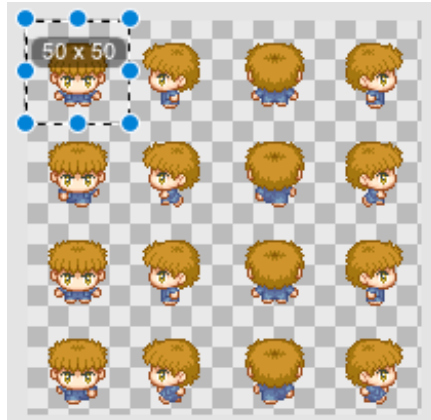


Figure 28. Player sprite sheet. Source: <https://opengameart.org/>

The size of the image is 192×192 . The frame size is 50×50 . I will use only two columns of frames, the second column will be used when player will run to the left, the last column for running to the right. The focus will be made on the IDs of each sprite. For example, for animation of the left running I will take sprites with IDs 1,5,9,12.

Another sprite sheet is needed to add some animation for enemies. I will use only first three sprites to implement the effect of running.



Figure 29. Enemies sprite sheet. Source: <https://opengameart.org/>

4.5 Implementation

This chapter is mainly focused on coding of a platformer game using HTML5, JavaScript, Phaser framework.

4.5.1 Files structure

The project folder consists of the following files:

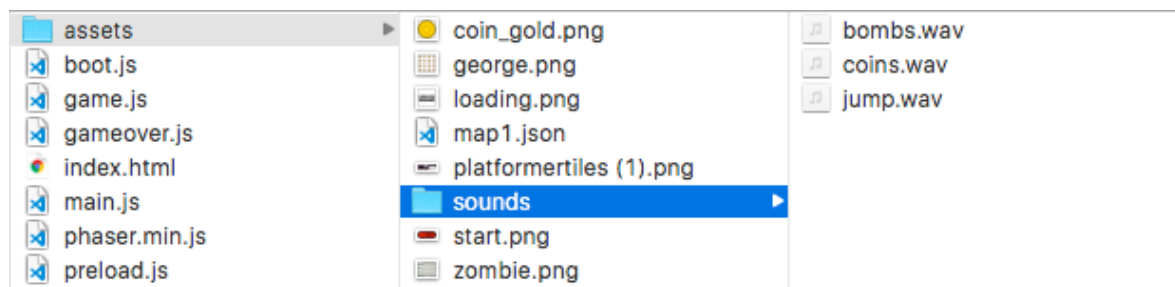


Figure 30. Project folder. Source: own processing

Lets take a look of index.html. All JavaScript files must be attached to it.

```
<script type="text/javascript" src="phaser.min.js"></script>
  <script type="text/javascript" src="boot.js"></script>
  <script type="text/javascript" src="preload.js"></script>
  <script type="text/javascript" src="game.js"></script>
  <script type="text/javascript" src="gameover.js"></script>
  <script type="text/javascript" src="main.js"></script>
```

One of the main feature of a game development in Phaser is a state controlling. States provide a possibility to separate a code on logical blocks. In the way that a game will change from one state to another. During writing a code I separate every independent block, like main block, menu, game to different states. I saved each state in a separate file: boot, preload, game, gaveover. It is not a requirement, but an advice to keep code more organized. All states are stored in main.js. States declaration is made by `game.state.add()`:

the first argument is the name of the state, while the second is the name of the function to call inside such state.

```
game.state.add('Boot', Boot);
game.state.add('Preload', Preload);
game.state.add('Game', Game);
game.state.add('GameOver', GameOver);
game.state.start('Boot');
```

So, the first runs state “Boot”, that loads the first screen, the welcoming, and the button “Start”, that starts the preloaded state. Preload state – loads all the assets.

4.5.2 Physics

There are 3 main physics engines in Phaser:

1. Arcade - bounded rectangles, without rotation. We draw a rectangle around our sprite and that will be our bounding box.
2. Ninja - bounded rectangles with rotation. The main difference with Arcade is that we can rotate our rectangles and place them in an angle. For example, if we want to create an up or down slope in our game level.
3. P2 - full-body physics with constraints, polygon support. This is the most advanced physics engine and is used for the more advanced/gravity-like games such as Angry Birds.

In the game I user Arcade physics type, as it is a platformer game type.

Set a physics for a whole game:

```
game.physics.startSystem(Phaser.Physics.ARCADE);
```

For every entity, a type of physics need to be specified. To enable physics the following code can be applicable:

```
enemies.physicsBodyType = Phaser.Physics.ARCADE;
game.physics.enable(player, Phaser.Physics.ARCADE);
```

To add gravity to game to make objects fall down.

```
game.physics.arcade.gravity.y = 300;
```

4.5.3 Player character

The tasks of the player are:

- to walk on game environment
- to jump
- to collect coins, by colliding with them
- to kill enemies, by jumping on them

To control a player walk, arrow keys were selected. Phaser has a built-in Keyboard manager and one of the benefits of using that is this handy little function:

```
cursors = game.input.keyboard.createCursorKeys();
```

This populates the cursors object with four properties: up, down, left, right, that are all instances of Phaser.Key objects. Then all we need to do is poll these in our update loop:

```
if (cursors.up.isDown)
{
    canJump = player.body.onFloor();

    if (canJump) {
        jumpAudio.play();
        player.body.velocity.y = -500;
    }
}

if (cursors.left.isDown)
{
    player.body.velocity.x = -250;
    player.animations.play('left');
}
else if (cursors.right.isDown)
{
    player.body.velocity.x = 250;
    player.animations.play('right');
}
```

The player increase score, while collecting coins. That solved with collision between player and Object Layer “Gold”. Under update function we add:

```
game.physics.arcade.overlap(player, coins, this.collectCoin, null, this);
```

The method “collectCoin” calls, after the collision between player and coins was detected. Audio effect is played, collided coin removed from the screen, added one more score, updated string with a new score.

```
collectCoin: function (player, coin) {
    coinsAudio.play();
    coin.kill();
    score += 1;
    scoreText.text = scoreString + score;}

```

The same concept works with enemies. But another function is called “livesMinus”. The player also can kill the enemy by jumping on it, its mean that we need to find on which part of the rectangle player and enemy collided. So, if we find out that player jumped on enemy, the enemy object will be removed from the screen.

```
if(enemy.body.touching.up && player.body.touching.down){
    enemy.kill();
}
```

Another situation is when player accidentally collide with the enemy, so the enemy hints the player. In that case, lives of the player will decreased by one, and if there is only 1 live and the enemy hinted the player, so the game is over state is starts.

```
if (lives > 1){
    enemy.kill();
    enemyAudio.play();
    lives -= 1;
    livesText.text = livesString + lives;
}

else if (lives == 1){
    lives -= 1;
    player.kill();
    this.newGame();
}
}
```

The methos newGame() just starts a gameOver state

```
this.state.start('GameOver');
```

4.5.4 Debug

Phaser has debug constructor, a collection of methods for displaying debug information about game objects. To debug the game that is running in Canvas mode, need to invoke all of the Debug methods from the game render function. They are drawn directly onto the

game canvas itself, so calling any debug methods outside of render they are likely to be overwritten by the game itself.

```
render: function() {  
    game.debug.body(player);  
    game.debug.bodyInfo(player, 16, 500);  
}
```

The result of the debug of the player object is shown on the picture below:



Figure 31. Debug mode on. Source: own processing

5 Results and Discussion

The result of this Bachelor Thesis is a platformer game. The application is uploaded on GitHub <https://github.com/vpukha/Platformer-game> , so everyone can use it, and not only for playing, but also for improving. The application is fully working, according to the requirement that highlighted before in this work. Of course, there are a lot of thing that could be added, to make the game more exciting. In this chapter, I would like to provide some point about the future development of the application.

5.1 Future development

The first point is to extend the game world. On this stage, there is only one level of the game, so the idea is to add more levels, 5 or more. The possibility to go to the upper level will be available, after some tasks will be done. The level can be completed on gold, silver or bronze medals, depends on time that was spent on a task.

The next idea is to make the game available on smartphones. It means two tasks: to make the application responsive to mobile view, and to provide some input controls, so the user can play just touching the screen.

The third thing to implement is a multiplayer. It is mean to add server-client architecture. The client is written in JavaScript with Phaser and runs in browser of the player. The server that could be used is Node.js and Express module. The Socket.io will provide communication between client and server.

6 Conclusion

During my Bachelor Thesis work I have described the main technologies needed for a 2D game development. In my thesis, I found new features of HTML5, compared to its previous versions, that makes the process of game creation easier for programmers. The main drawing APIs are: Canvas, SVG, WebGL. The work follows with description of game engines. I divided all game engines into two main groups: the visual and the scripting one. The most popular visual game engine Construct2 is very useful for people who just start programming. The engine is very easy in use, but it costs money, depends on team numbers, profitability of the product, etc. I compared 6 JavaScript frameworks according to criterias that fit my needs in creation of a game in my practical part. After the comparison, I chose Phaser framework I found that it is actively maintained, supports third-party tools like Tiled, Texture Packer, and others, it is easy to deploy, and it provides a developer support services. Speaking about the functionality it has built in three physics engines: Arcade, Ninja, P2. So, basically there is no need to think about coding a physics, but to focus on other things, like logic and behavior of the game process.

In my practical part, I described, the process of game implementation using JavaScript technologies, mainly using Phaser framework. I found it very interactive to use framework in creating a HTML5 game. First of all, because of built-in functions, that works perfectly. Of course, it is not a problem to make the same game without any framework, using just JavaScript, the only difference will be in lines number of code, and also a time that is spent on creation.

The output of a bachelor thesis is a platformer game. The technologies that were used in development are HTML5, JavaScript and Phaser framework. Tiled Map Editor was selected to draw a map of a gaming environment. The Google Chrome browser with its Developer Tools were used during development for testing and debugging the application.

7 References

- (1) **SHANKAR, Aditya Ravi.** *Pro HTML5 Games*. New York : Springer Science Business Media New York. 978-1-4302-4711-1.
- (2) **MacDonald, Matthew.** *HTML5: The Missing Manual*. First Edition. Sebastopol : O'Reilly Media, Inc., 2011. ISBN: 978-1-449-30239-9.
- (3) **JONATHAN Robie, Texcel Research.** What is the Document Object Model? *Level 1 Document Object Model Specification* . [Online] 07 20, 1989. [Cited: 12 20, 2017.] <https://www.w3.org/TR/WD-DOM/introduction.html>.
- (4) **FULTON Steve, FULTON Jeff.** *HTML5 Canvas*. Second Edition. Sebastopol : O'Reilly Media, 2013. 978-1-449-33498-7.
- (5) **WILLIAMS, James L.** *Learning HTML5 game programming*. First Edition. Boston : Pearson Education, 2011. ISBN 978-0-321-76736-3.
- (6) **Clay.io.** HTML5 Game Engines. [Online] [Cited: 01 05, 2018.] <http://html5gameengine.com/>.
- (7) **SUBAGIO, Aryadi.** *Learning Construct 2*. First edition. Birmingham, UK: Pack Publishing, 2015. ISBN: 978-1-78439-767-8.
- (8) **Scirra Ltd.** Choose Your Plan. *Construct*. [Online] [Cited: 01 05, 2018.] https://www.construct.net/cz/make-games/buy-construct-3?utm_source=scirra&utm_medium=homepage&utm_campaign=homepage&utm_content=topbuynow.
- (9) **Scirra Ltd.** Construct 2 System Requirements. *Construct*. [Online] 01 16, 2018. <https://www.scirra.com/manual/6/system-requirements>.
- (10) **GOSE, Stephen.** *Phaser.js Game Design Workbook*. Independently Published, 2016. ISBN: 978-1520154558.
- (11) **Awio Web Services LLC.** Browser and Platform Market Share. *W3Counter*. [Online] 12 01, 2017. [Cited: 12 20, 2017.] <https://www.w3counter.com/globalstats.php>.
- (12) **Microsoft Corporation.** IntelliSense. *Microsoft Corporation*. [Online] [Cited: 12 20, 2017.] <https://code.visualstudio.com/docs/editor/intellisense>.
- (13) **CC by 3.0.** Attribution 3.0 Unported. [Online] [Cited: 01 23, 2018.] <https://creativecommons.org/licenses/by/3.0/>.