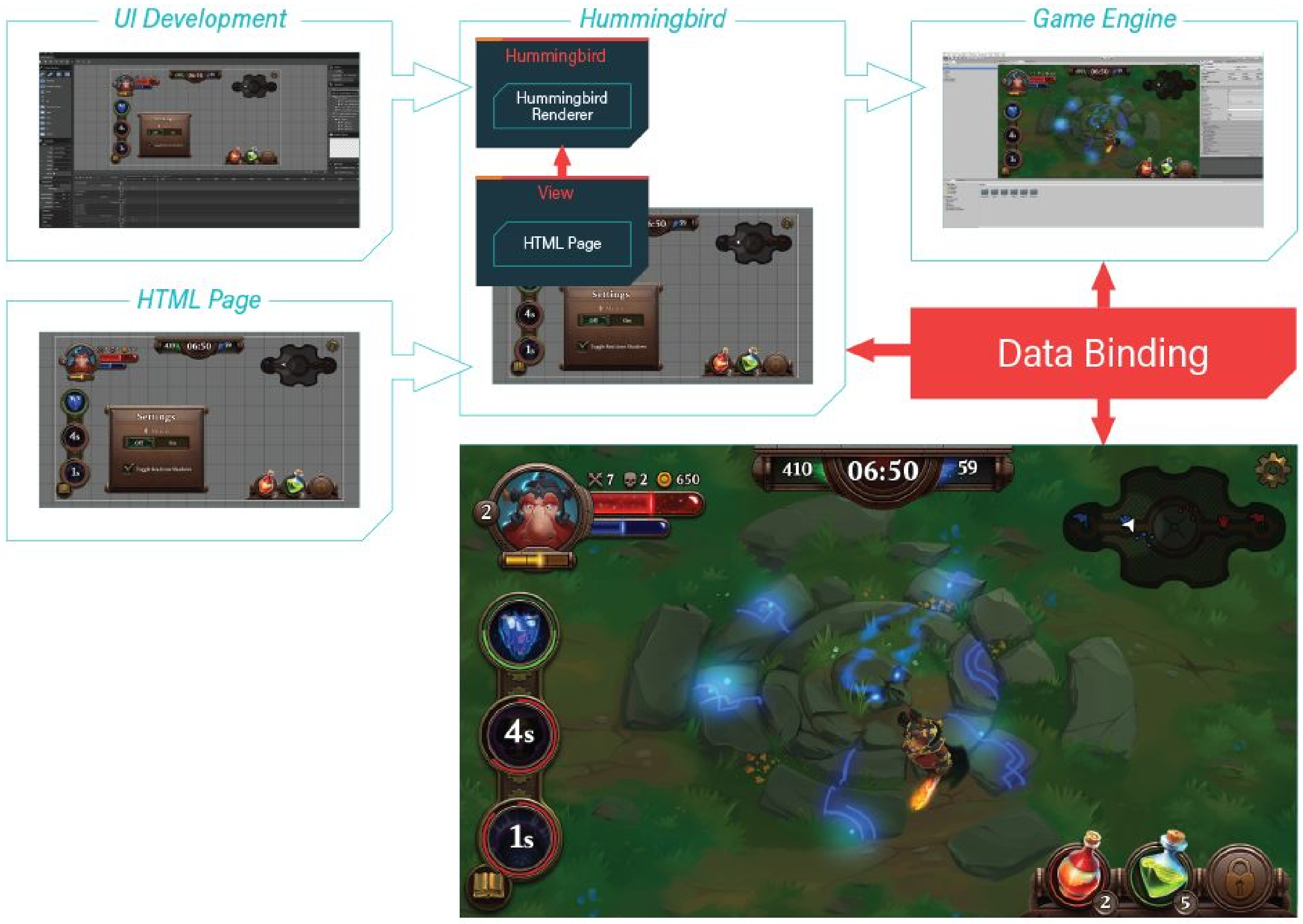
About this document

The purpose of this document is to guide you through the UI development process. It goes over all the major phases and describes the software and technologies used. Please note that this a general workflow guide. For detailed documentation of Hummingbird please refer to the **Hummingbird** Documentation file or the documentation section on coherent-labs.com.

Introduction to Hummingbird

Hummingbird is a graphical user interface system specially designed for mobile games. Game developers and UI artists can use standard modern HTML, CSS, and JavaScript to design and implement game UI and create great user experience.

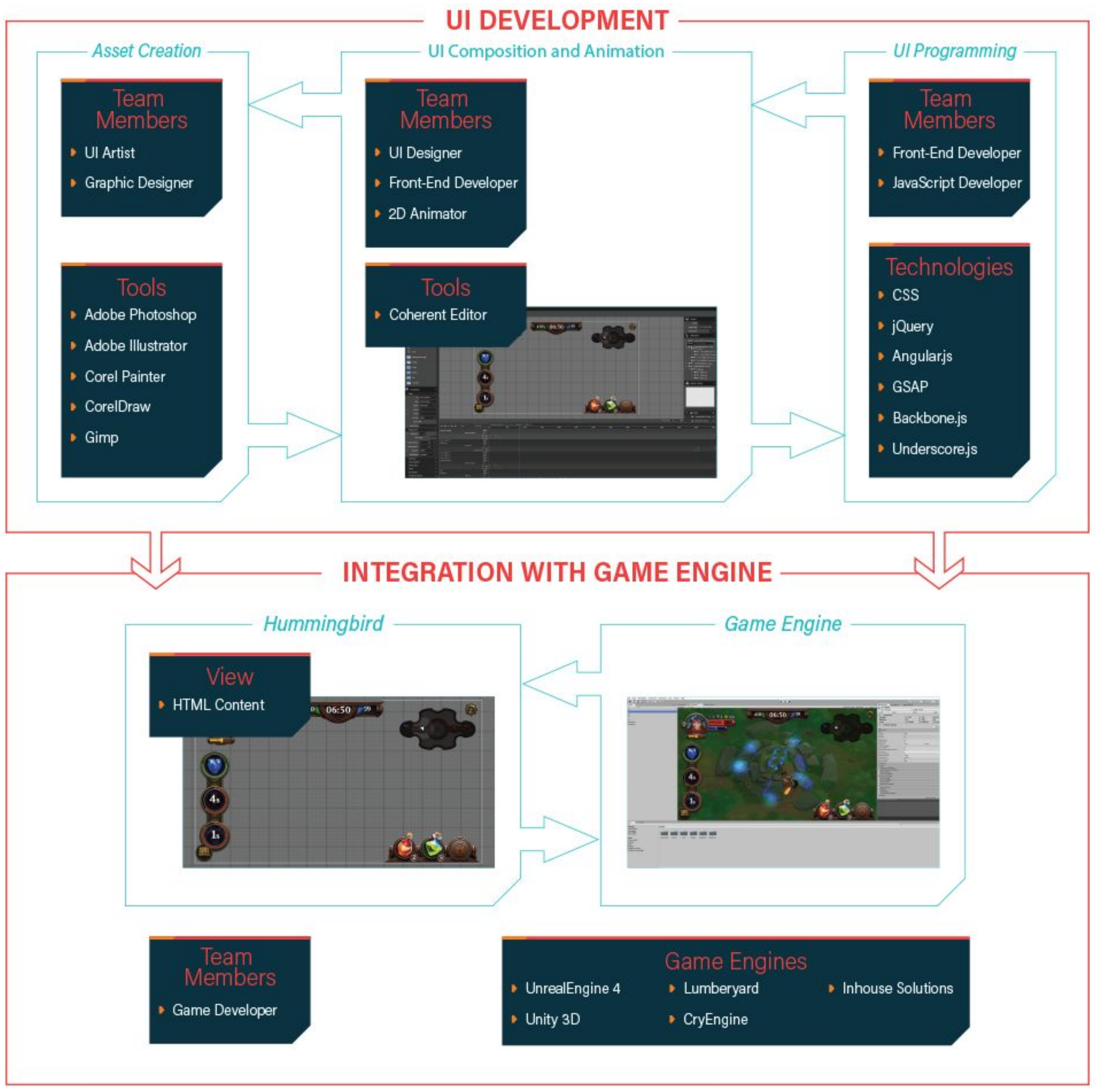


Above you can see the basic functional diagram of **Hummingbird**. Essentially, the result of the **UI development** is a HTML page that is placed inside of **Hummingbird View**. The **Hummingbird View** is a HTML5 page and the JavaScript context for it. The **View** allows you to perform various operations on the page, such as resizing, navigating to a different URL, sending input events, executing custom JavaScript code and so on. The HTML content in the **View** is rendered by the

Hummingbird Renderer to a texture that is passed to the **game engine** which displays it in the **game**.

UI development

Major phases and tasks distribution in UI development



The HTML UI development process can be broken down in three major tasks – **UI vector and bitmap assets** design, **UI composition and animation** and **UI programming**. All three tasks can be carried out in parallel and the result is HTML page/pages that are displayed in **Hummingbird view/views** that are integrated in the game engine. The tasks are discussed in greater detail further on in the guide but in the diagram above you can see the team member involved and the software used for each phase.

UI vector and bitmap graphics assets creation

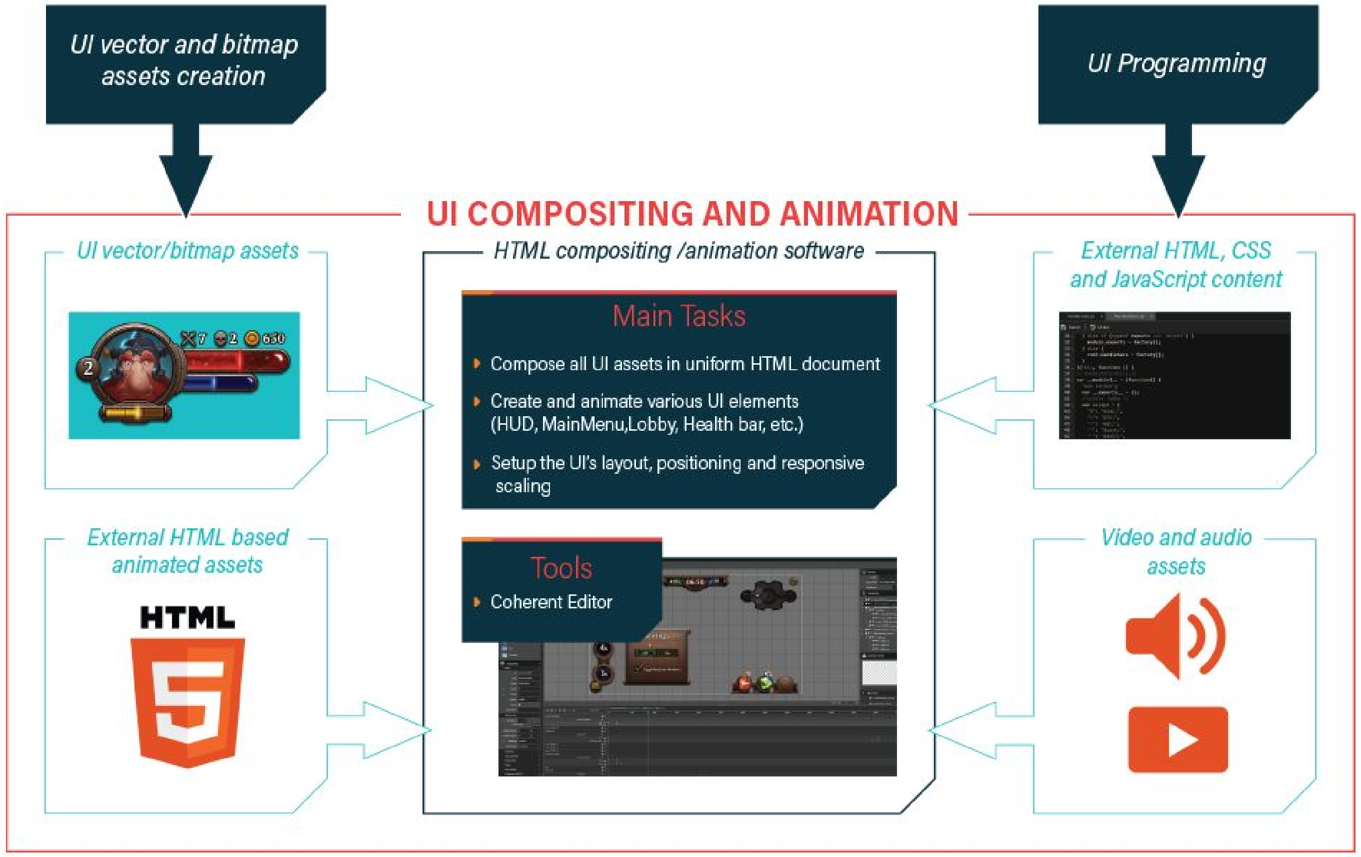
Although, you can create basic shapes such as circles, rectangles, ellipses directly with HTML elements, very often you will need more complex shapes for the UI. These shapes are typically created as **bitmap** (raster) or **vector** images by UI artist or a graphic designer.

Bitmap (raster) graphics images represent the image as rectangular grid of pixels. Bitmap images are used for all sorts of UI elements: artwork, photos and detailed UI elements. Their major drawback is that their quality degrades if they are scaled.

Vector images are based on geometrical primitives and scale perfectly regardless of the size of the UI. Also, the file size of the generated files is smaller than that of raster images.

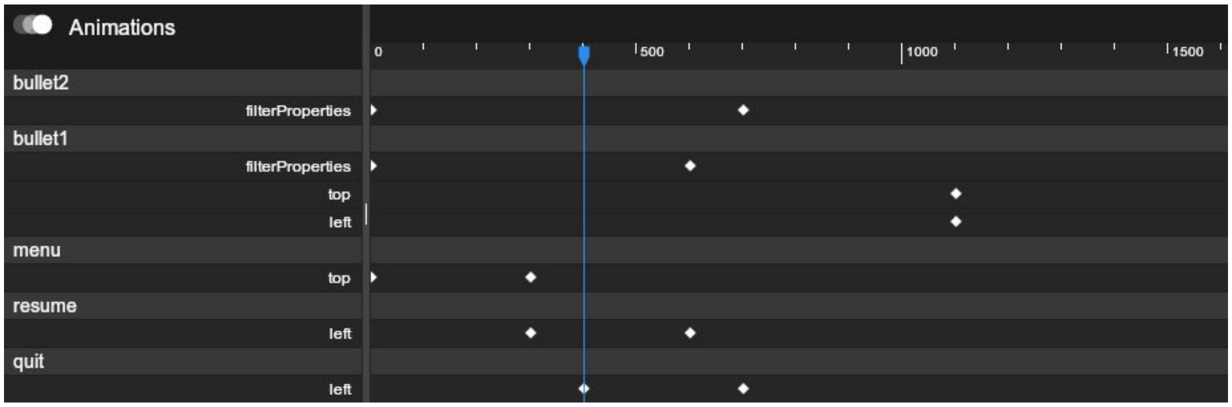
Hummingbird supports **SVG vector** images that can be modified by CSS/JS at runtime (check the SVG images section for more information). The major drawback of vector images is that they are actually rendered at runtime and complex images might affect the performance. **It is advisable to use vector images only when the UI has to be scaled for large resolutions or when you design simple shapes.** On the diagram above you can see the software used for the different types of UI image assets.

UI composition and animation

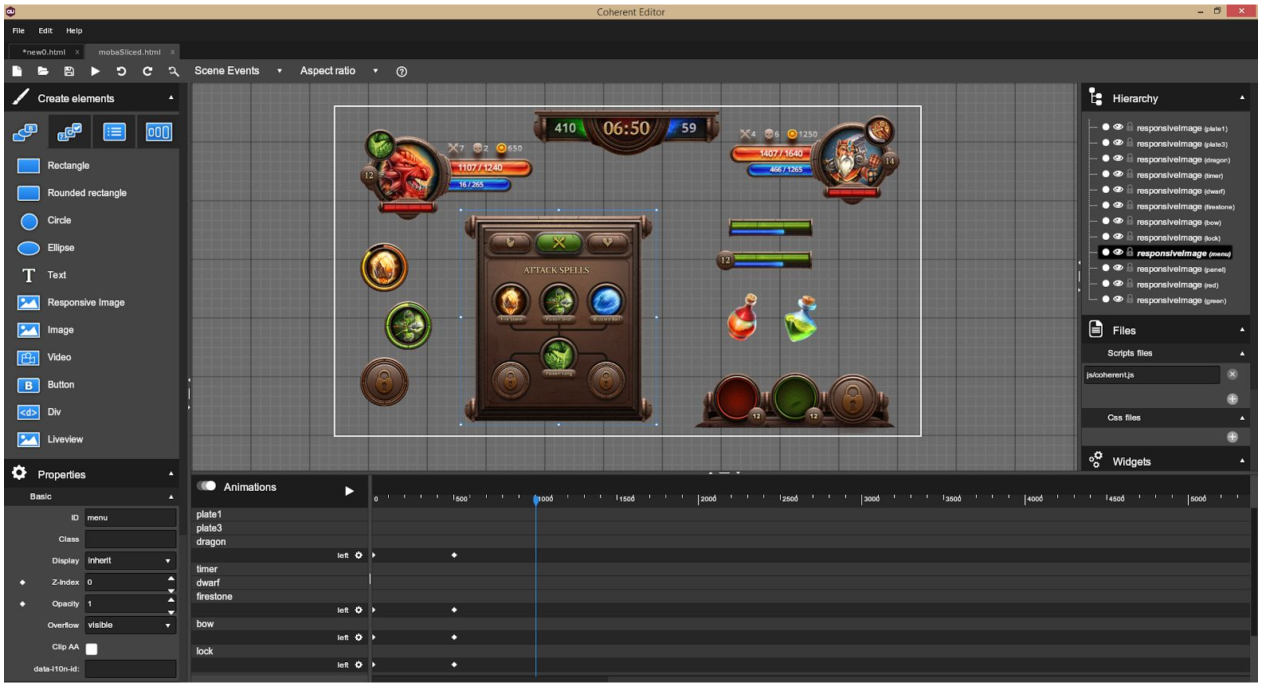

UI vector and bitmap assets creation
UI Programming
UI COMPOSITING AND ANIMATION
UI vector/bitmap assets
HTML compositing /animation software
External HTML, CSS and JavaScript content
Main Tasks
Compose all UI assets in uniform HTML document
Create and animate various UI elements (HUD, MainMenu,Lobby, Health bar, etc.)
Setup the UI's layout, positioning and responsive scaling
External HTML based animated assets
HTML
5
Tools
Coherent Editor
Video and audio assets

This stage can be carried out by the **UI designer**. It has three main tasks:

- **Compose all the UI assets** (vector and raster images, video, audio, external HTML/CSS/JS) in a uniform HTML documents. With **Hummingbird** this step is very easy. Simply export all UI assets from Photoshop as a HTML page and import them in the Coherent editor. The assets will keep their original position.
- **Create and animate various UI elements** (HUDs, Menus, Lobby, Health bars, progress bars, etc). Use the visual timeline animations panel in the Coherent Editor to add various animations to the UI elements.

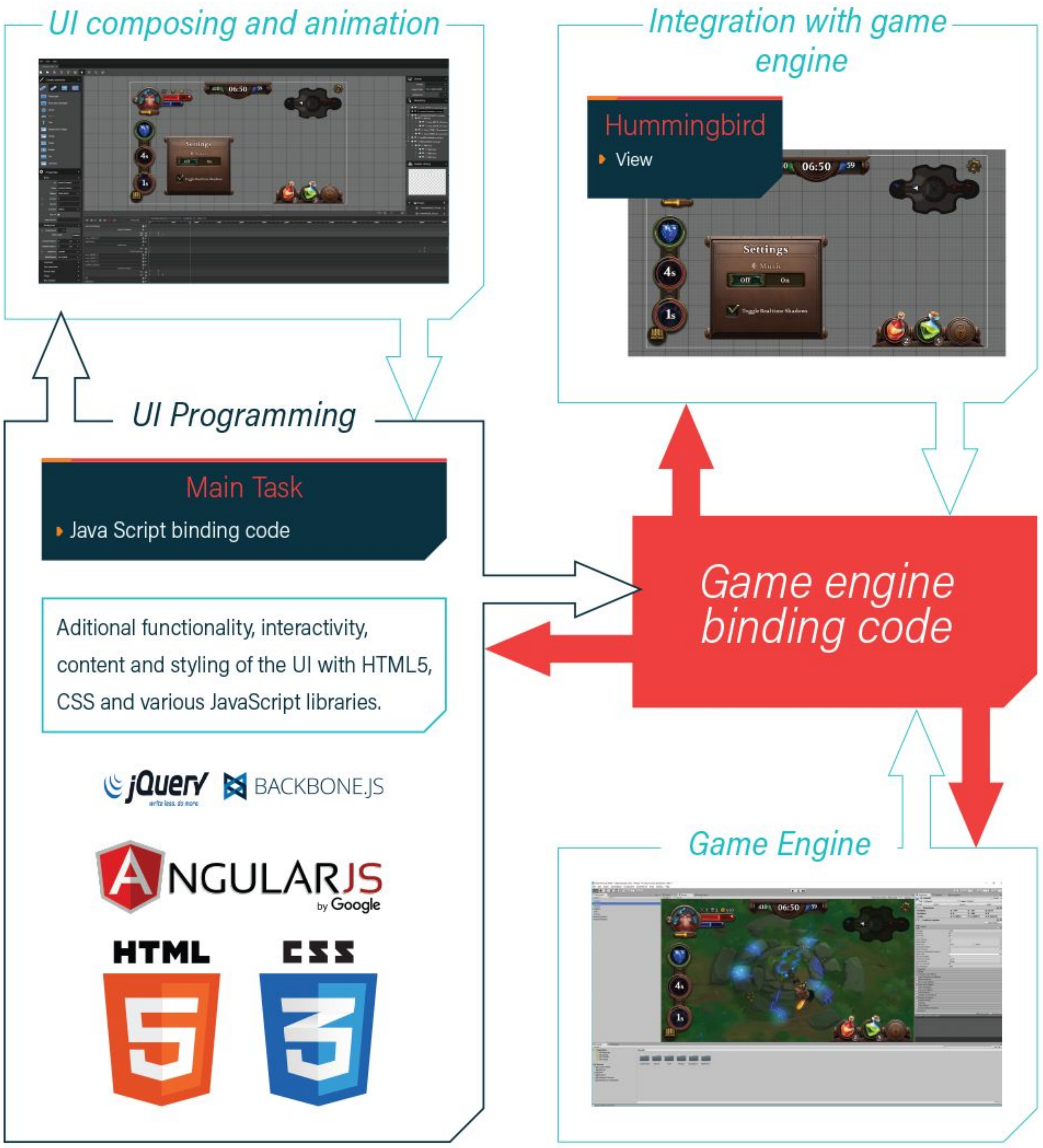


- **Setup the UI layout**, positioning and responsive scaling.



*The **Coherent Editor's** quick start guide is available here. Of course using the **Coherent** Editor is optional. In this stage you can also use any standard HTML editor.*

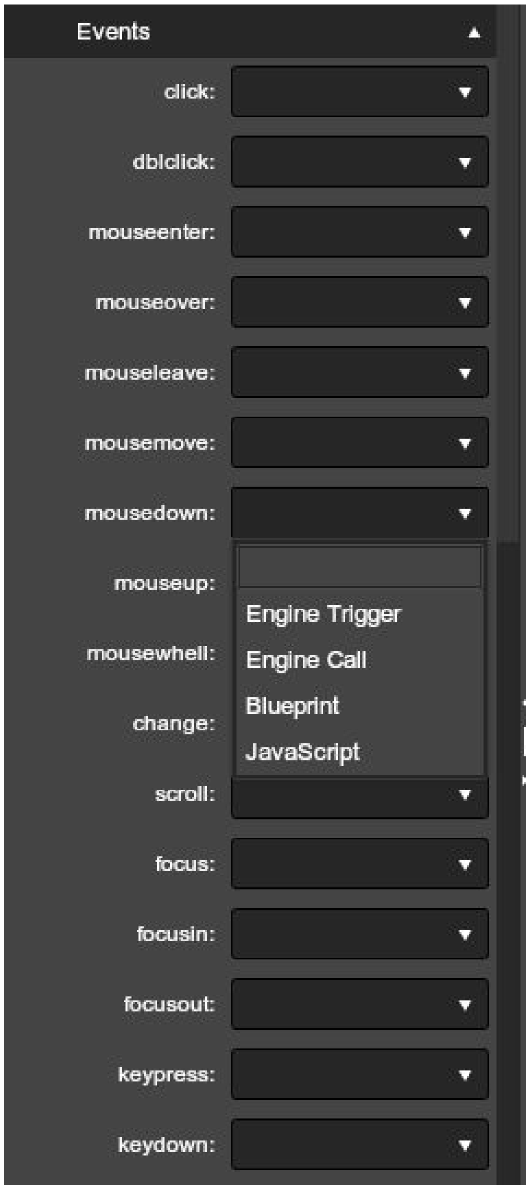
UI programming


UI composing and animation
Integration with game engine
Hummingbird
View
UI Programming
Main Task
Java Script binding code
Aditional functionality, interactivity, content and styling of the UI with HTML5, CSS and various JavaScript libraries.
jQuery
BACKBONE.JS
ANGULARJS by Google
HTML
CSS
Game engine binding code
Game Engine

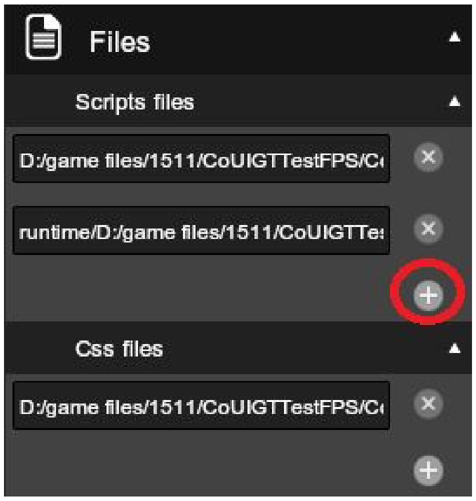
This stage is very closely related to the UI composition and animation stage. You can easily do it with the **Coherent Editor**.

It has three main tasks:

- Set up the JavaScript part of the code for the data binding through the Editor **events panel**

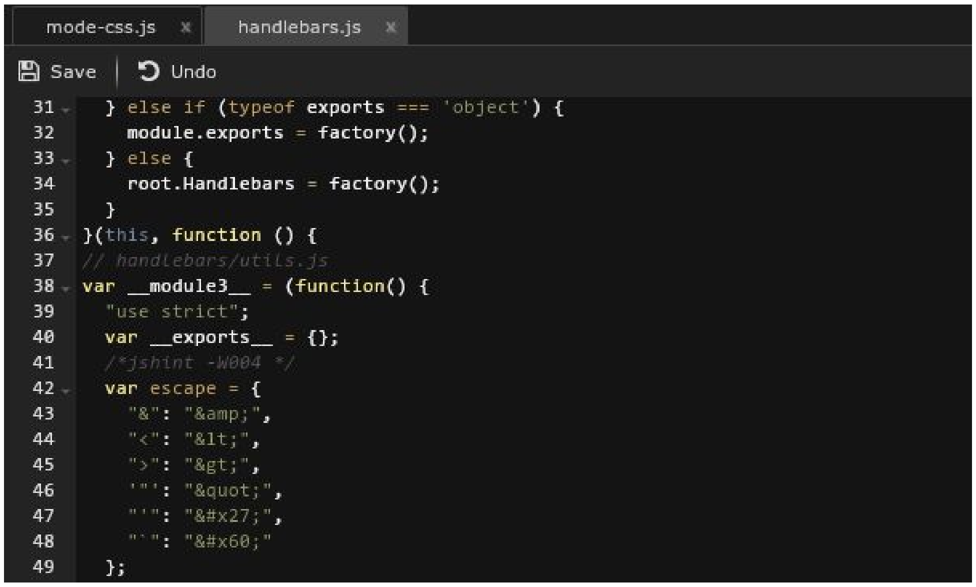


- Add additional functionality, interactivity, content and styling of the HTML pages by using HTML5, CSS and various JavaScript libraries. Additional JavaScript libraries or CSS files can be imported in the **Files panel**.


Files
Scripts files
D:/game files/1511/CoUIGTTestFPS/C
runtime/D:/game files/1511/CoUIGTTe
Css files
D:/game files/1511/CoUIGTTestFPS/C

- Add additional animations with the **Hummingbird JavaScript animation framework**

All of these task can be performed by the UI designer via the visual editor. If you are more comfortable with writing your own JavaScript, you could use the Coherent Editor code editor.



```
  } else if (typeof exports === 'object') {
    module.exports = factory();
  } else {
    root.Handlebars = factory();
  }
}(this, function () {
// handlebars/utils.js
var __module3__ = (function() {
  "use strict";
  var __exports__ = {};
  /*jshint -W004 */
  var escape = {
    "&": "&amp;",
    "<": "&lt;",
    ">": "&gt;",
    '"': "&quot;",
    "'": "&#x27;",
    "`": "&#x60;"
  };
```

At this stage you can use the Hummingbird Debugger in collaboration with a game developer to debug for any HTML/CSS/JS errors.
/The debugger is still in development

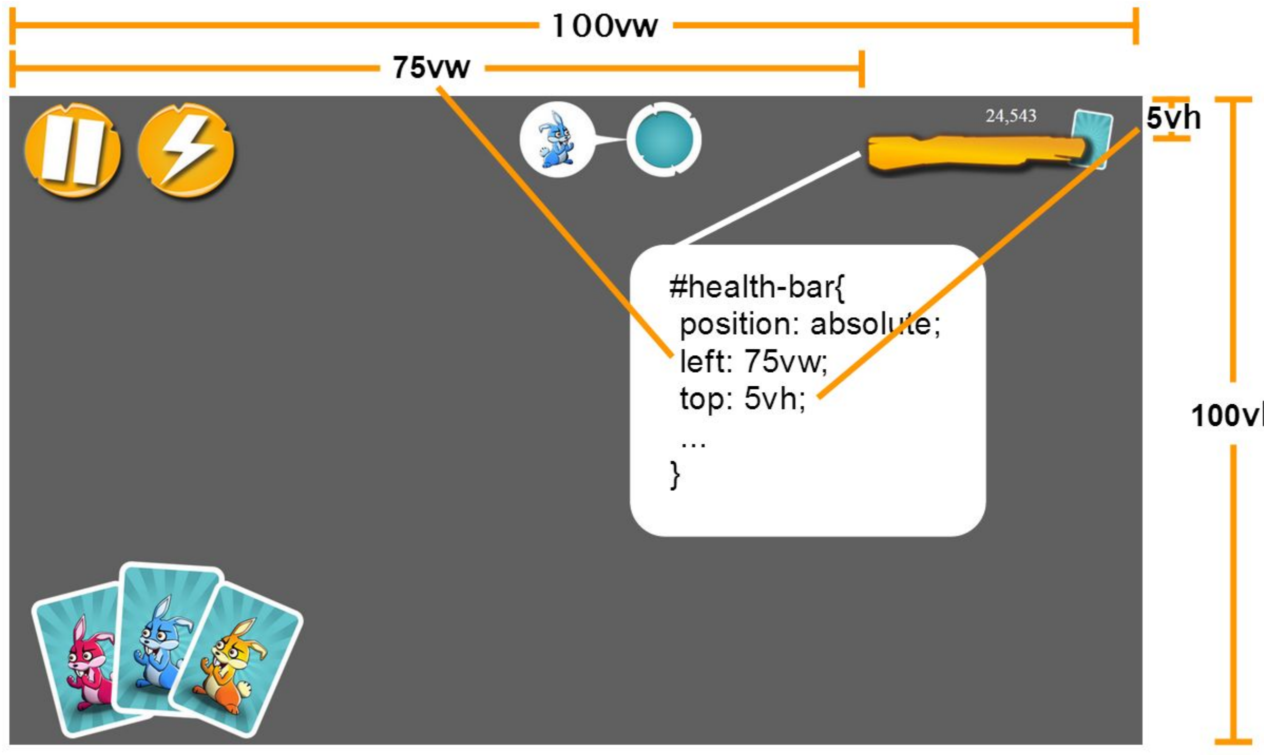
Integration with a game engine

This phase is typically carried out by a game developer. He is responsible for adding the **Hummingbird View** component and creating its settings in the game engine.

Additional UI development tips

Creating responsive UI

Web technologies offer many different approaches that you can use to position and scale the UI. However, in most of the cases the best approach is to use viewport units. Viewport units (vw and vh) can be used both for the position and size -vw is 1/100th of the rendered viewport width and vh is 1/100th rendered viewport height.
These units makes the UI responsive, properly scaled and positioned regardless of the screen resolution. The position can be anchored using properties such as "top", "bottom", "right" and "left" while the size of the elements defined by "width" and "height" properties.



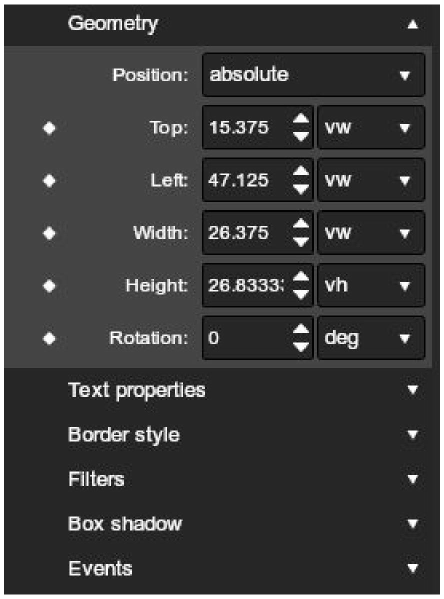
```
#health-bar{
 position: absolute;
 left: 75vw;
 top: 5vh;
 ...
}
```

For example, here are the styles for the plank that represents the health bar in the sample above:

```
#health-bar{
   position: absolute;
   left: 75vw;
   top: 5vh;
```

```
    width: 21vw;
    height: 11vh;
    background-image: url('img/healthBarFull.png');
    background-size: contain;
    background-repeat: no-repeat;
}
```

It has "left" property set to "75vw" and "top" property set to "5vh". This will effectively make the health bar appear at 75% of the viewport width and 5% from the viewport height. Vmin and vmax unitas also could be used. Vmin units are 1/100th of the smallest side and vmax units that represent 1/100th of the largest side. One possible problem with viewport units however is that if the HTML elements are nested, their position and size could be scaled relative to their parent and not to the viewport. To achieve this, you can use percent units(%). They work the same way as viewport units but instead of 1/100 of the viewport size they represent 1% of the size of the parent element. /percent units are still in development/


Geometry

Position:	absolute	
Top:	15.375	vw
Left:	47.125	vw
Width:	26.375	vw
Height:	26.8333:	vh
Rotation:	0	deg

For more information about viewport, percent as well as other CSS units check the following article - http://www.w3schools.com/cssref/css_units.asp

Different styles for different screen sizes

In the cases that you should have visually different UI for different screen size, you should use CSS media queries. Youjust need to specify the range in which these styles should apply using "max-width" and "min-width".

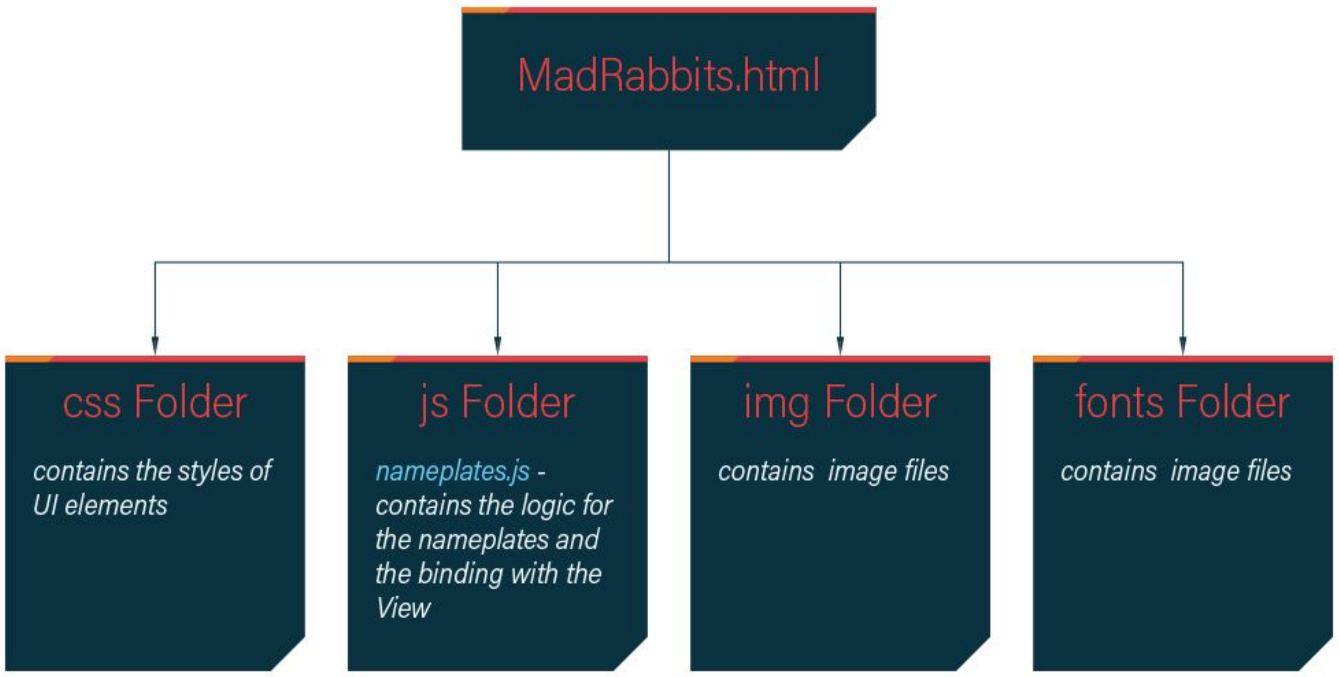
```
@media all and (max-width: 699px) and (min-width: 520px) {
  #sidebar ul li a {
    padding-left: 21px;
    background: url(../images/email.png) left center no-repeat;
    }
}
```

For more information about media queries check the following article - https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Modular UI

Sometimes, several team members could be working on the same HTML file. For example, one UI developer could be working on a minimap of a HUD, while another could be working on the healthbar.

This could be easily done with the **Hummingbird Editor**. Each UI developer can work on one part of the UI and then export it as a widget. Then the chief developer just needs to import all the individual widgets in the main UI composition.


MadRabbits.html
css Folder
contains the styles of UI elements
js Folder
nameplates.js - contains the logic for the nameplates and the binding with the View
img Folder
contains image files
fonts Folder
contains image files

Improving the JavaScript code with Google Closure compiler

Google Closure compiler (https://developers.google.com/closure/compiler/) is a tool for making JavaScript download and run faster. Use it to reduce the size of the JavaScript code, check for errors and improve the overall performance of the UI.

It has three optimization levels:

The WHITESPACE_ONLY compilation level - removes comments from the code and also removes line breaks, unnecessary spaces, extraneous punctuation (such as parentheses and semicolons), and other whitespace.

The SIMPLE_OPTIMIZATIONS compilation level - performs the same whitespace and comment removal as WHITESPACE_ONLY, but it also performs optimizations within expressions and functions, including renaming local variables and function parameters to shorter names. Renaming variables to shorter names makes code significantly smaller. Because the SIMPLE_OPTIMIZATIONS level renames only symbols that are local to functions, it does not interfere with the interaction between the compiled JavaScript and other JavaScript.

The ADVANCED_OPTIMIZATIONS compilation level - performs the same transformations as SIMPLE_OPTIMIZATIONS, but adds a variety of more aggressive global transformations to achieve the highest compression of all three levels. The ADVANCED_OPTIMIZATIONS level compresses JavaScript well beyond what is possible with other tools. Make sure to check the error tab as it can reveal all sorts of programming errors that are not parse errors such as: using undeclared variable, unreachable code, re-assinging constants, etc. To improve the code one can also use code quality tools such as Google Closure Linter, jsLint and jsHint.