# Web Tech Report

* A for HTML
* A for CSS
* A+ for JS
* A for PNG
* A for SVG
* A for Server
* A for Database
* A+ for Dynamic pages
* ? for Depth

HTML: XHTML delivery, have gained a high level of confidence with structuring my pages. No frameworks used.

CSS: Have gained a lot of experience using it and making my pages look modern. No frameworks used.

JS: Can confidently implement any feature possible with it, featuring the dodgegame.  
The aim of the game is to not get hit by incoming bullets while destroying enemy blocks. There are 3 modes implemented, two use separate maps that are stored in files server-side and the third one is the endless mode, spawning enemies until the player loses. Performance is measured and submitted to the database. Level information on dynamic page section.  
Also used some javascript for the leaderboards page to query the database and update the html table.  
Initially used p5.js, but adapted all features to be pure javascript with no frameworks used.

PNG: Edited an image’s selected region by changing its color hue in a separate transparent-filled layer. Located on the About webpage. Used GIMP (Check next pages)

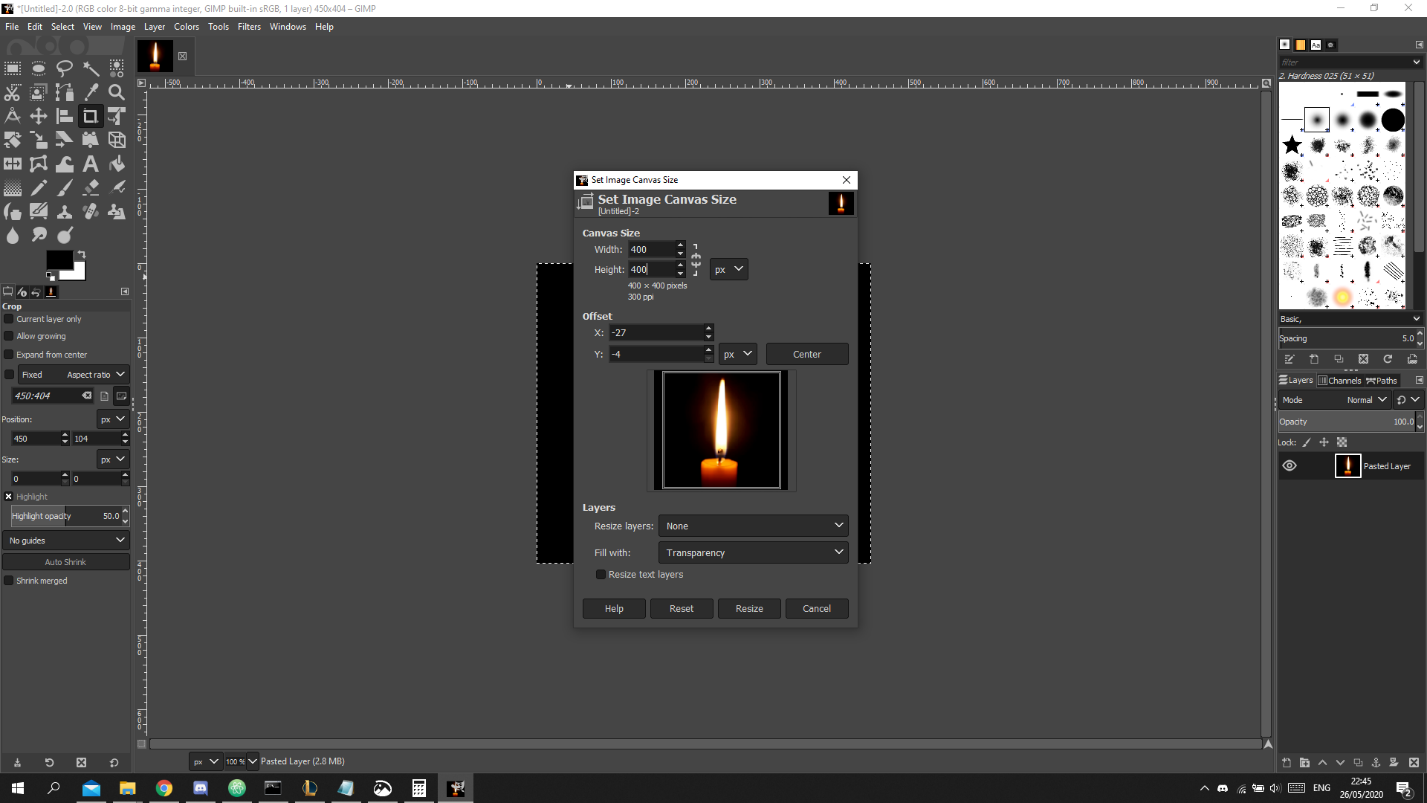
SVG: Path editing of objects, grouping them, transforming them to create the website’s vector logo. Used Inkscape (Check next pages)

Server: Configured URL validation and content negotiation amongst using https and an openssl certificate for a secure connection. Using port 8443 for development.

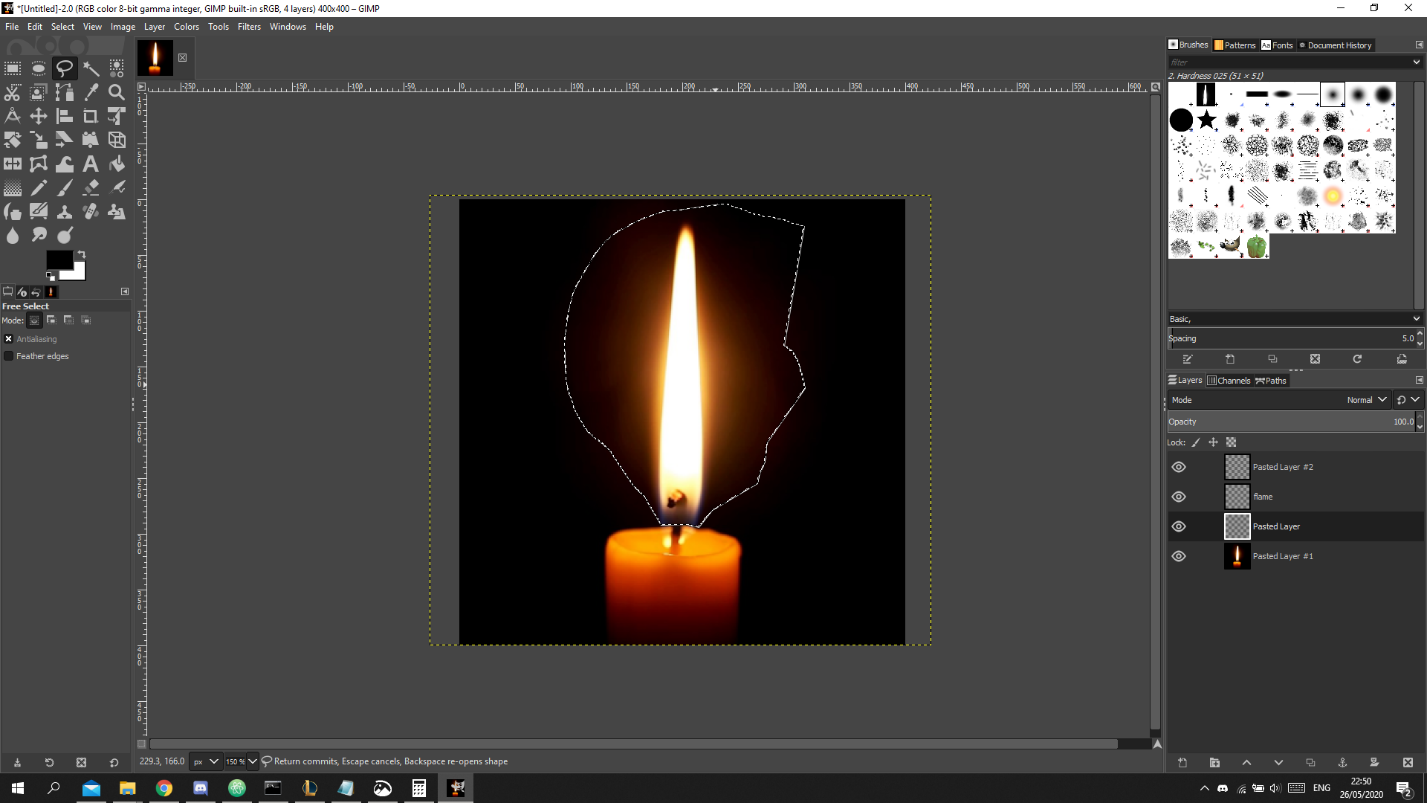
Database: Already familiar with SQL, created two tables in an sqlite embedded database holding information of each submitted game result, and the highest score of every unique user respectively. These values are updated with an HTML form on the index page using HTML POST and fetched with javascript on the leaderboards page.

Dynamic pages: The javascript game uses json files that load information about when and where to spawn an enemy, and what its power is (life, damage, firerate, reloadspeed, magazine size etc.) These are stored server side, the server automatically handles distinct URLs and sends the JSON file to be processed by the client. The leaderboards page works in a similar fashion, delivering database queries instead of a file’s contents.  
For failsafe, if a file does not exist server side, the default game mode of Endless applies.   
Level 1 is stored as gamedata1.json, Level 2 is gamedata2.json, and Level 3 demonstrates the failsafe with a garbage name (“gamedataasdf.json”) which doesn’t exist on the server and delivers Endless mode, a single line JSON specifying type Endless. The mode then operates client side to save bandwidth. Gamedata1 is loaded as default if the user doesn’t select a level and starts the game.

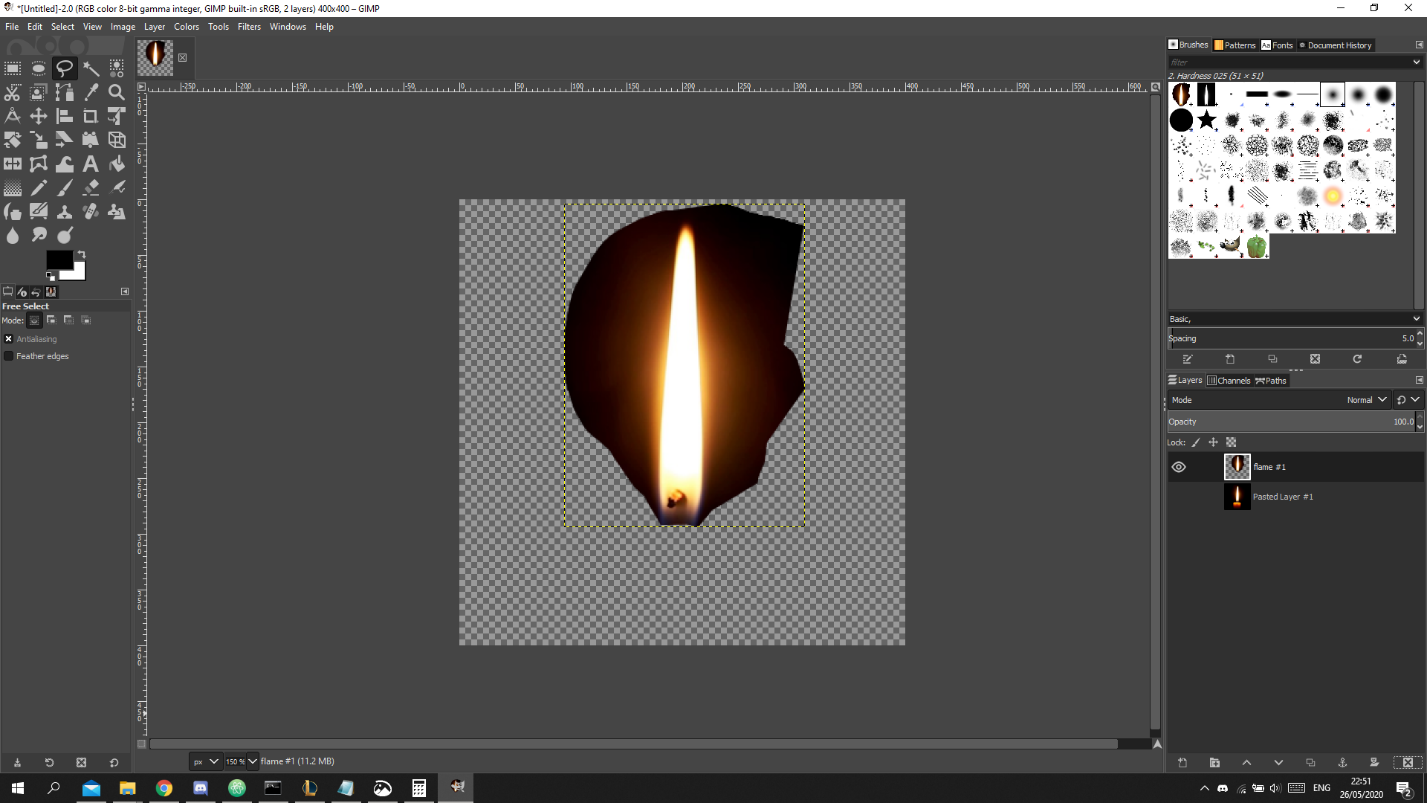
PNG(annotated procedure):



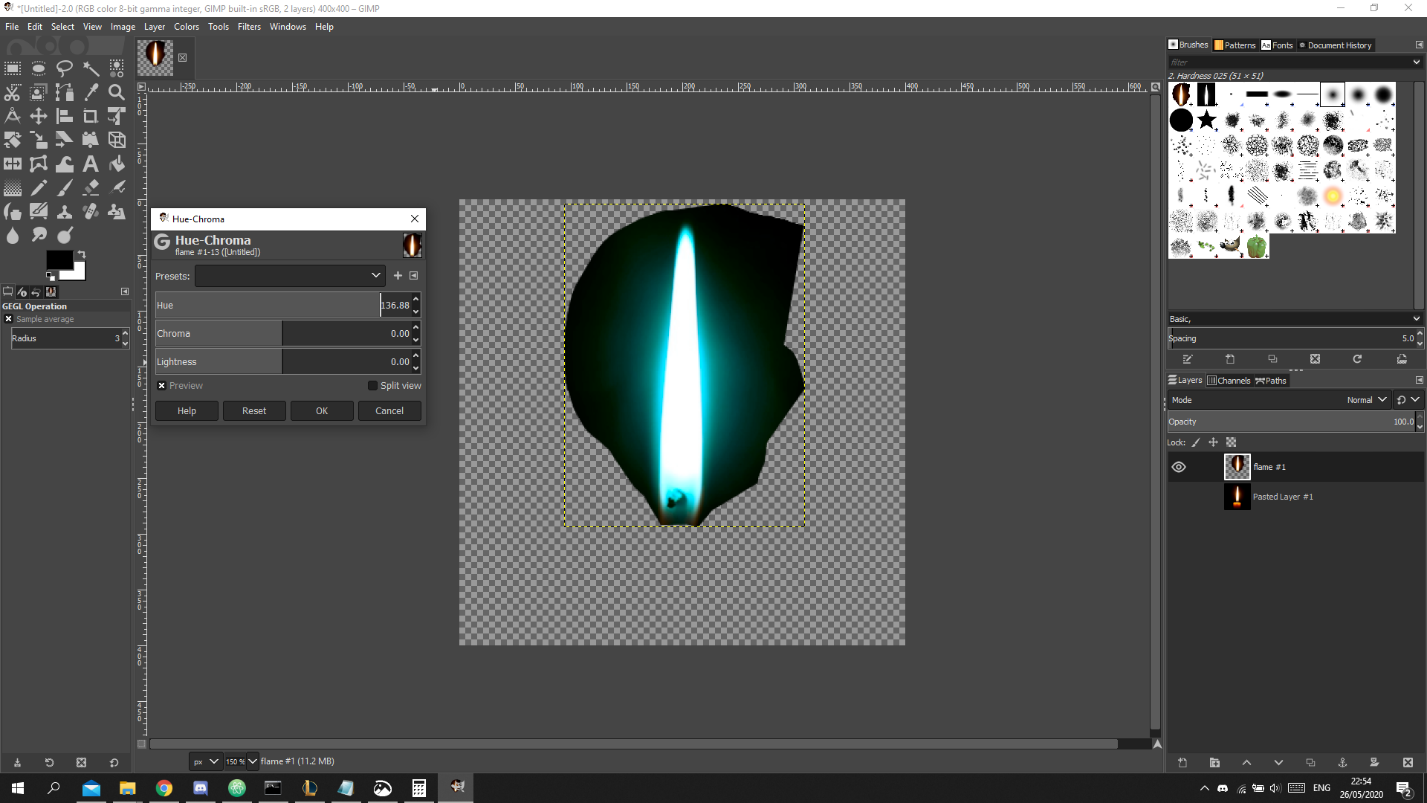
Crop the original photo to a neat size of 400x400px



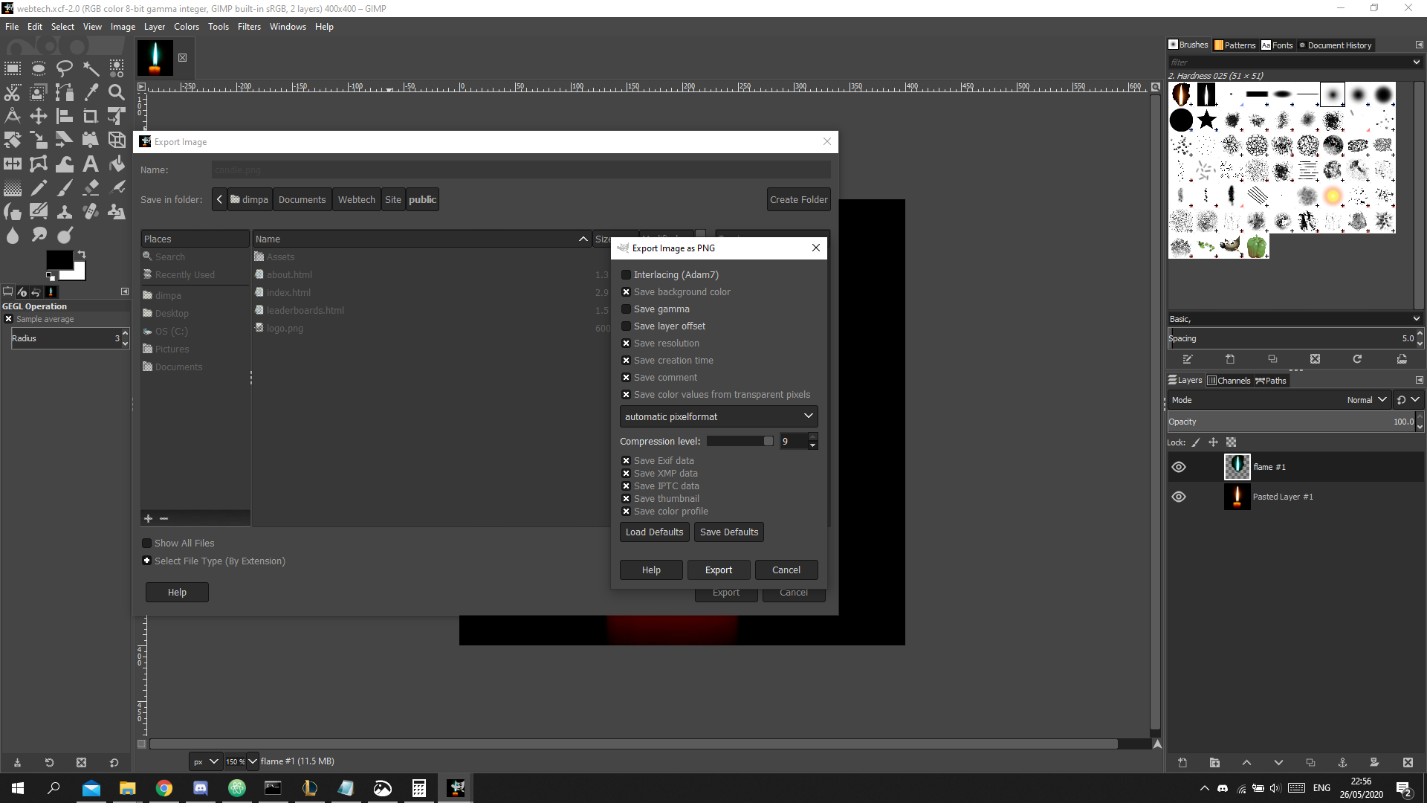
Using the free-hand tool, select the flame alone



Paste the selection onto a new transparent layer



Using the Colours-Hue-Chroma menu change the hue to a teal colour

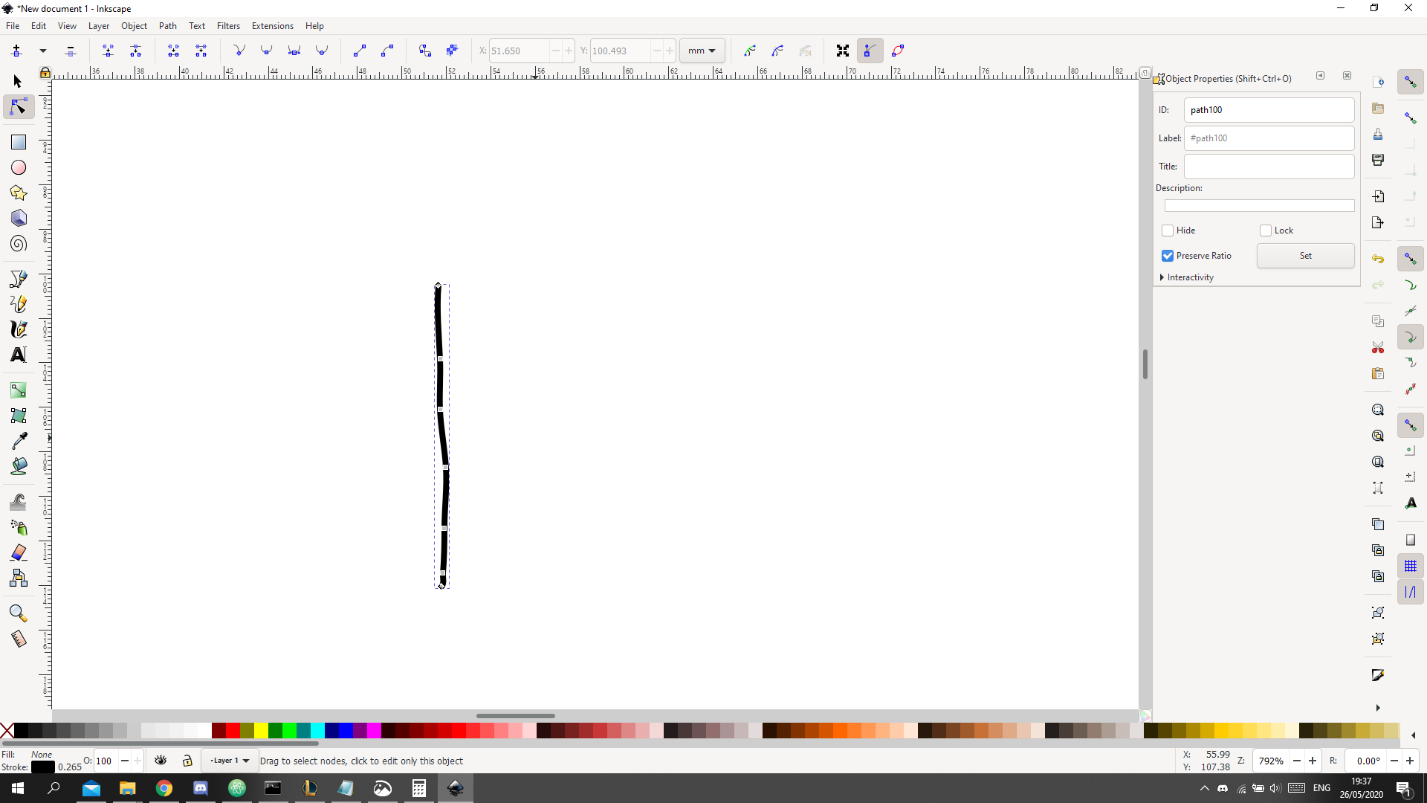


Make the base layer visible again and export to PNG (while also saving it as an xcf for record)

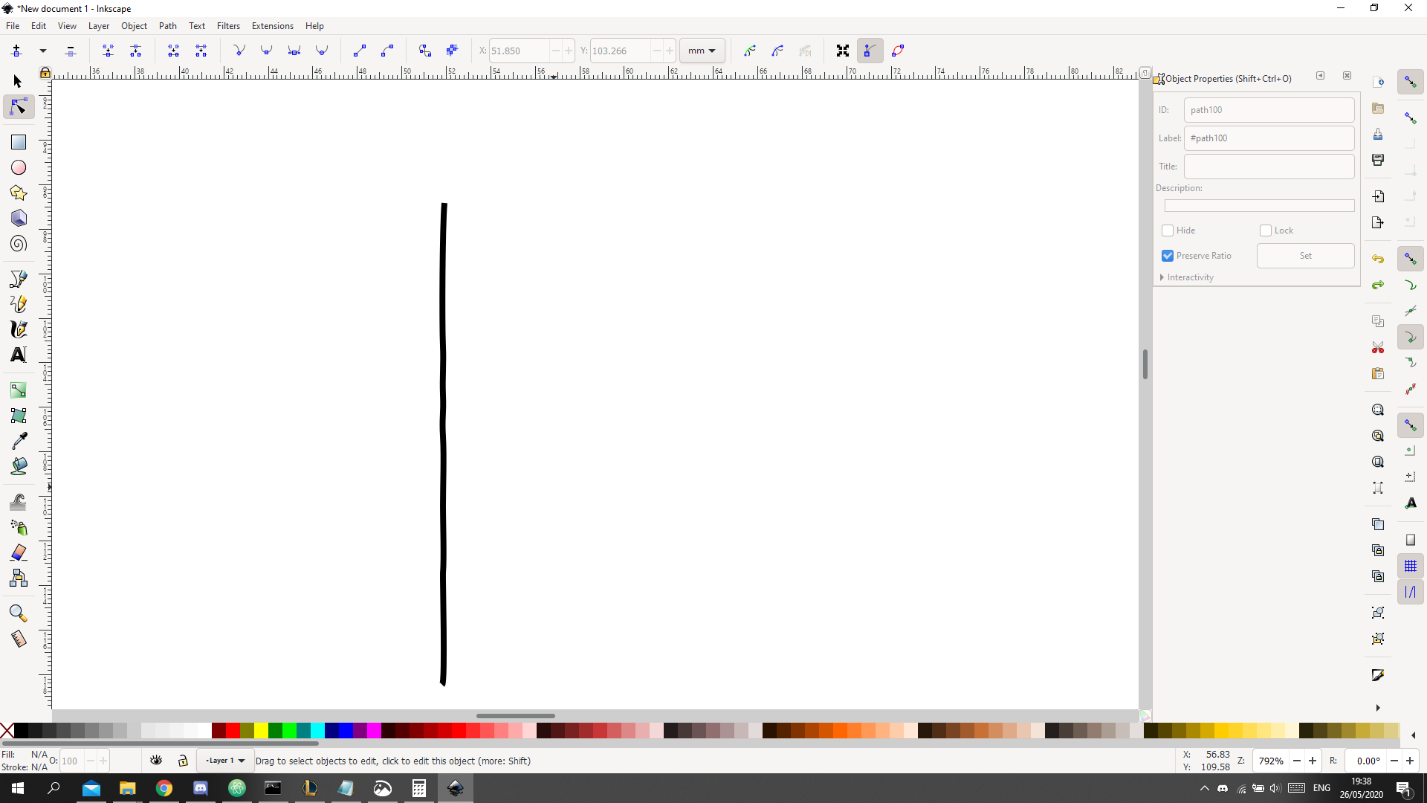


The final image!

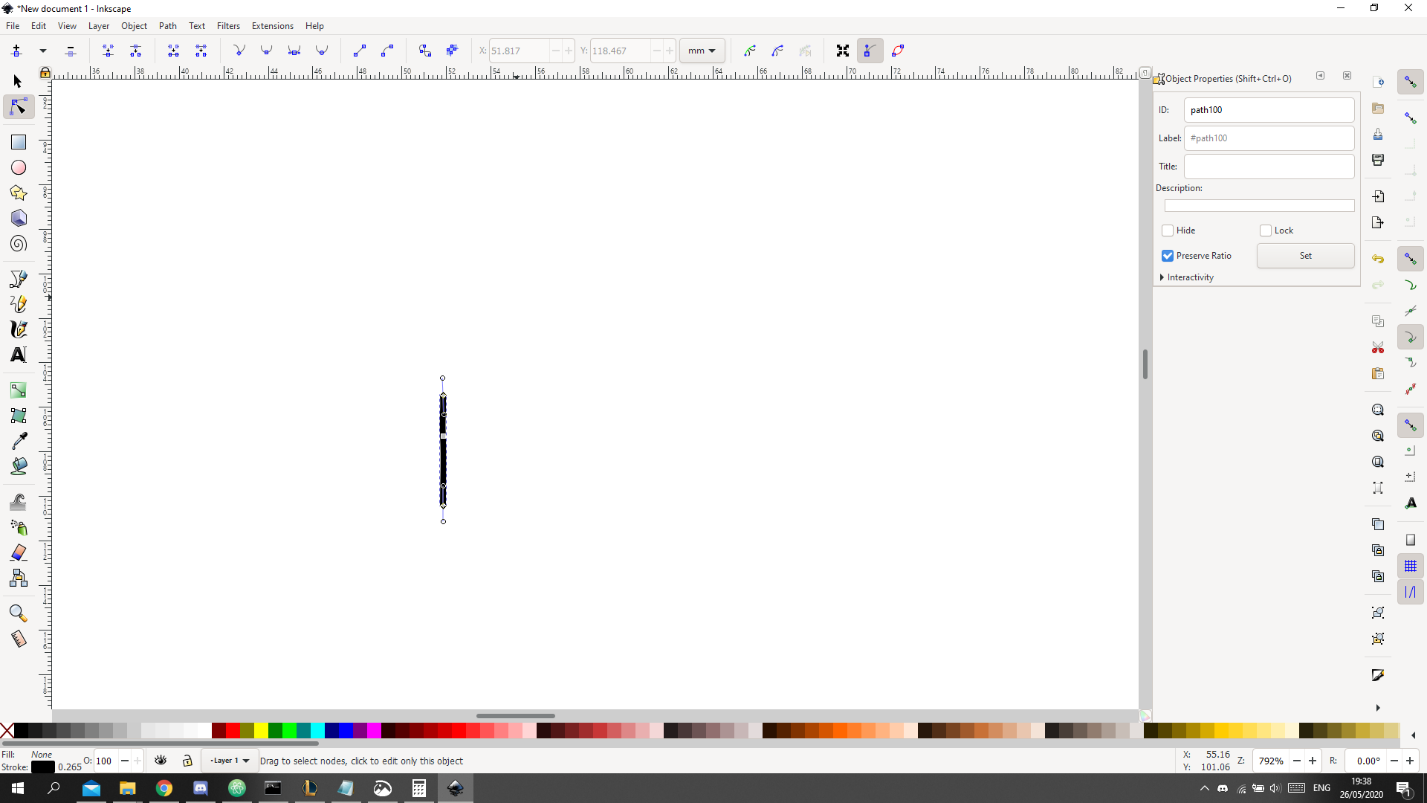
SVG (annotated procedure):



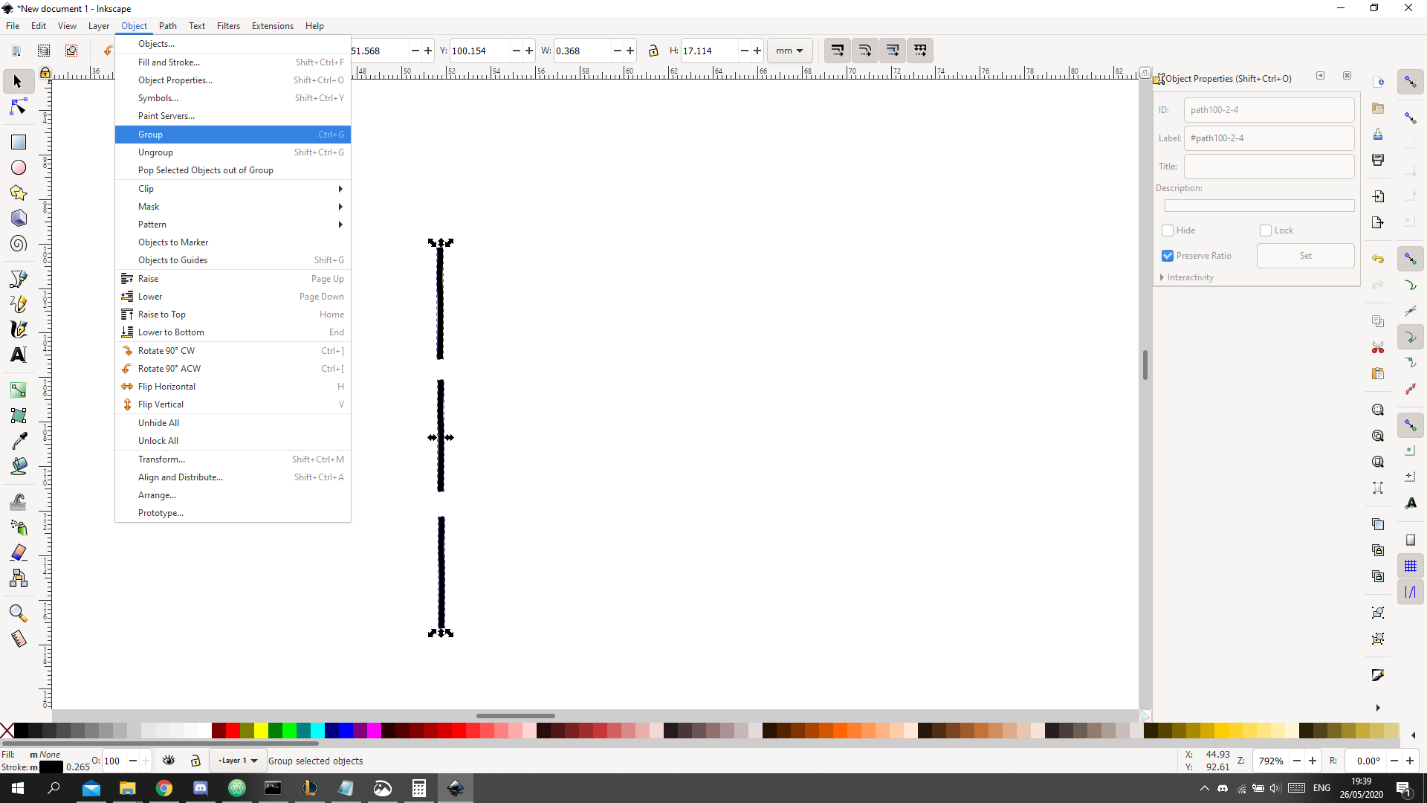
Use the free draw tool to make a line as best as possible



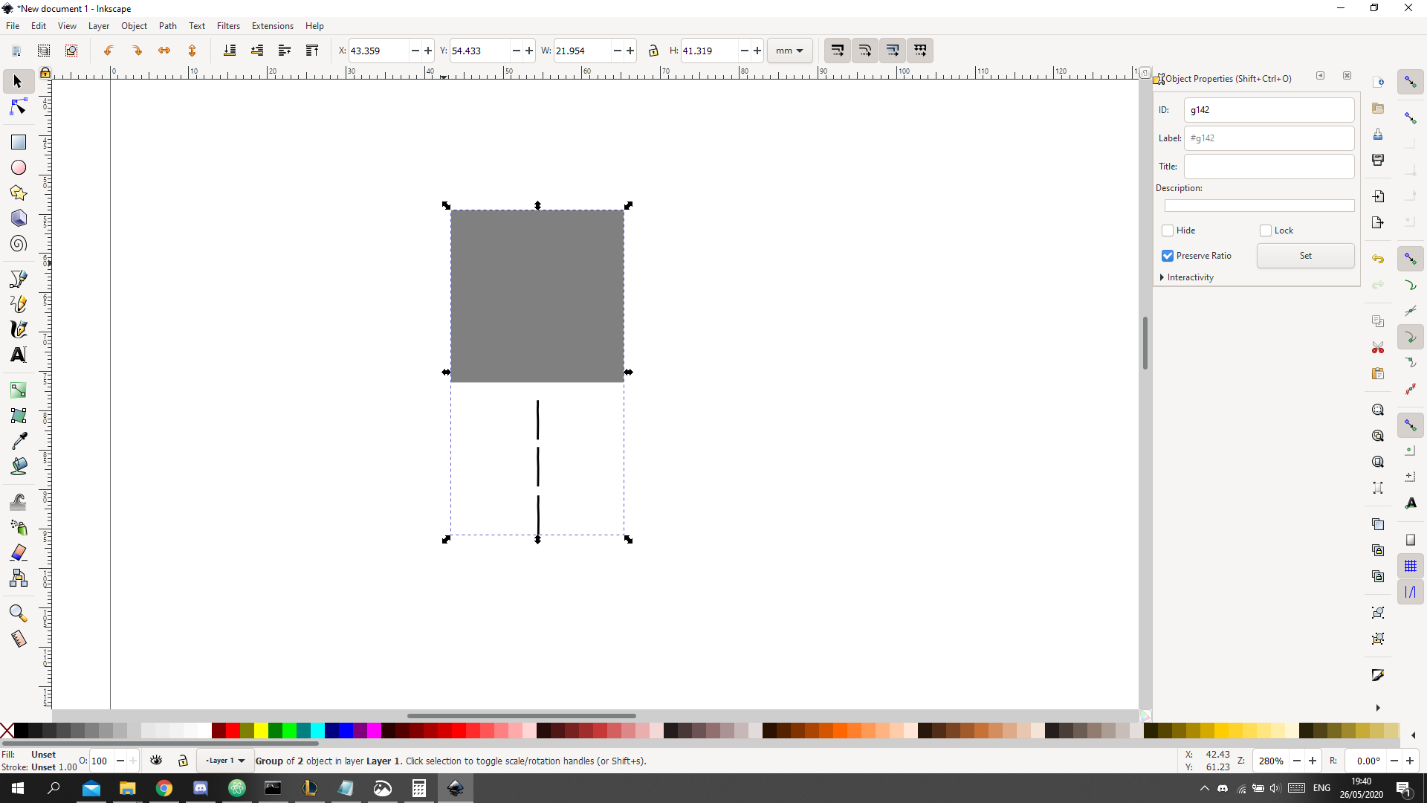
Use the path edit feature to straighten the line. Making it longer makes it easier from my experience



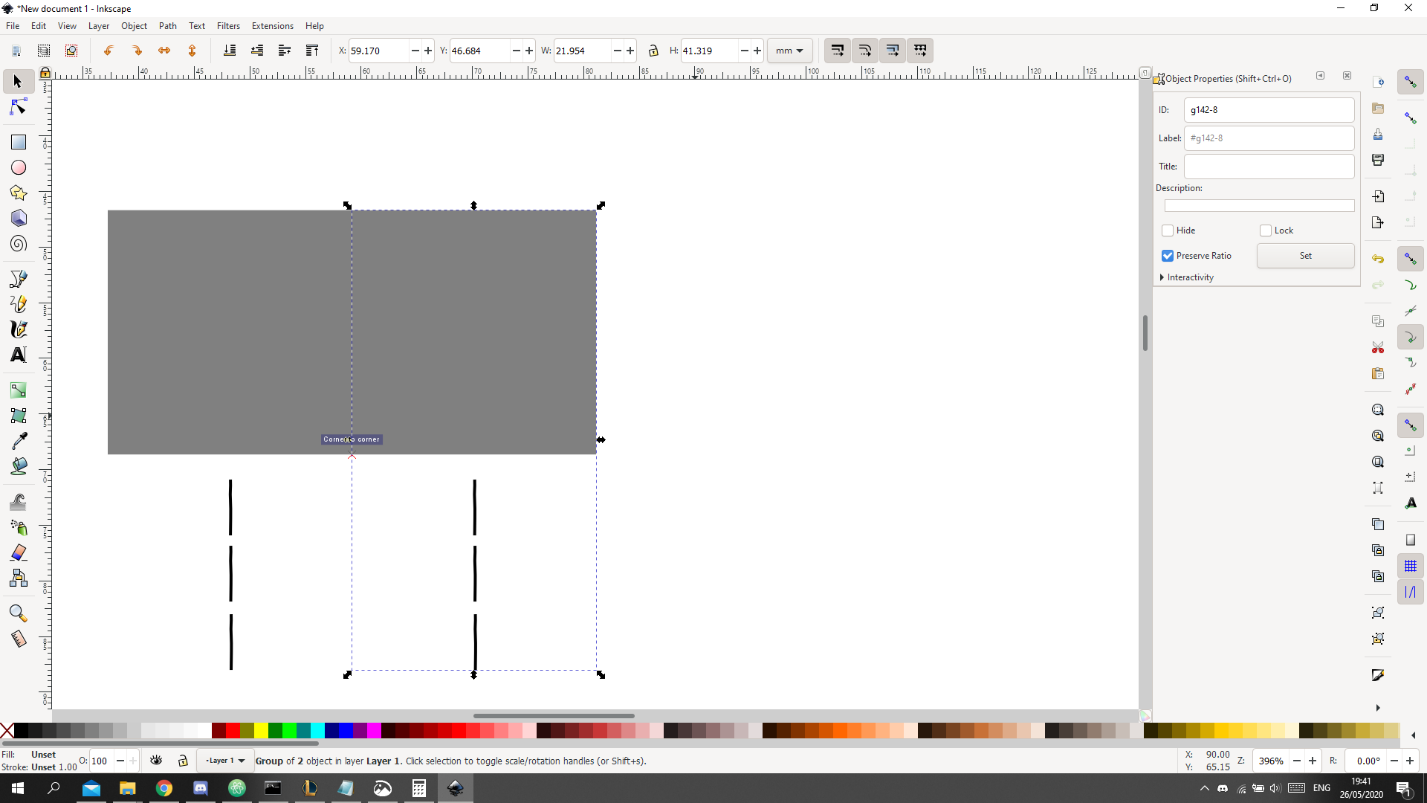
Select the long ends and cut them out to have a small bullet sized length



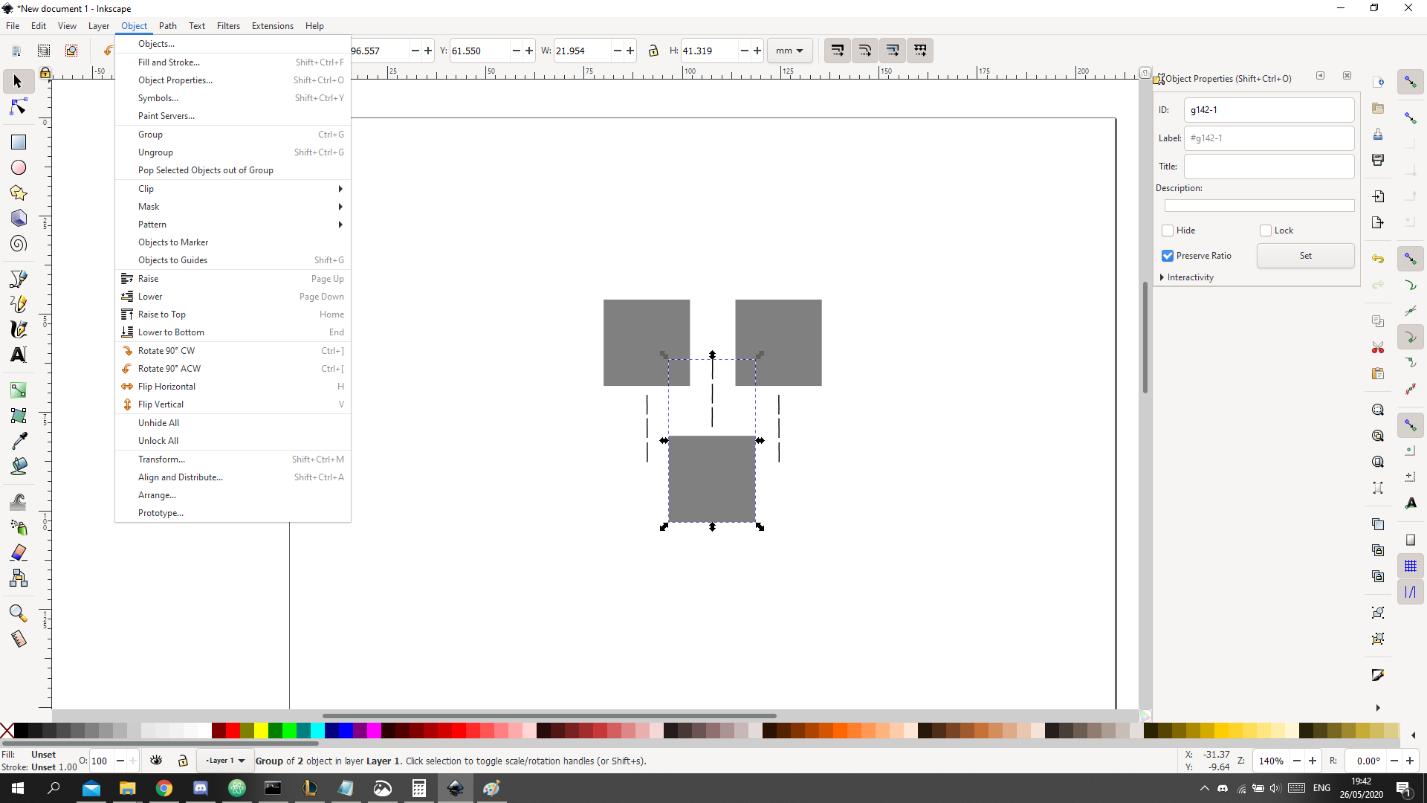
Duplicate the object and group them together to form a single bullet line



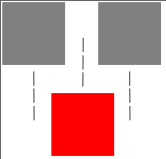
Hold control while dragging a rectangle to keep the aspect ratio intact, and group it with the bullets.



Duplicate it and line them up. Hold control to move it in a straight line



Duplicate and flip vertically. Finally, fill it with red color to match the game’s player object.



The logo!