Project overview.

This project is a collection of 9 mini-games. All of those games are designed in the same style. They are played in a specific order, with the 'rewards' from the first 8 games used in the last one. The games include riddles, puzzles, memory games, mazes, etc. Parts of the project are inspired by or reference certain films, well-known characters, music, or literature, which is reflected in the design.

Minigames.

In order of appearance in the project:

Dots – certain circles (stars) in the sky need to be connected. A counter is shown to the user (e.g. 5/10, then 5 more connections need to be made). The stars look identical to other non-connectable ones. However, if the user hovers above a connectable start, the cursor becomes a pointer and the star(s) it connects to change color (they go back to the original color once the mouse is moved).

Scream – a game show where the user is playing against 2 non-playable characters. There is a phrase displayed on the board and the players can guess letters to see where they appear (and eventually guess the entire phrase). The game includes a wheel that each player spins each turn to see how many points a correctly guessed letter will give them. There are other rules, rewards, and penalties in the 'game show', which are explained in detail by the 'host' character.

Film – a memory game with 12 pairs of scenes from classic (and not only) horror films. The cards are shuffled and shown face-down. The user can reveal 2 at a time and remember where they are to eventually pair everything up. They must do so with a limited number of 'reveals'.

Blackout – a board with clues is 'hidden' behind a black canvas with a transparent radial gradient where the user's mouse is. The gradient changes size based on audio from the user's microphone. The clues explain what needs to be done to continue to the next game. The main part of it also includes the microphone audio (a specific audio pattern, determined with comparison of the audio volume).

Map – the user is presented with 7 input boxes with numbers in them (number of letters in each needed words) and when hovering, an alphabetic cipher of the place's coordinates is shown. By deciphering them and writing the latitude, longitude, and zoom values into designated fields, the user sees the needed word(s) – the name of a certain place in a MapBox.

Bat – a version of Flappy Bird stylized for this project, with a background filled with frowning clouds and a bat flying between gravestones. To pass the game, the user (bat) must not collide with any of the gravestones for 90 seconds.

Labyrinth – includes 2 parts:

- 2D maze, where a ball can be moved through the maze using arrow keys, there are randomly distributed items within the maze, all of which must be collected prior to exiting
- 3D 'labyrinth', more specifically, a model of *Relativity* (M.C.Escher painting). There are multiple possible path segments (represented by videos made in Blander) which connect to each other, allowing the

user to choose the next direction with arrow keys (signified which ones on the screen)

Puzzle – after each previous game, more puzzle pieces would appear in the toolbar (seen in all games except *dots*). Those pieces can now be dragged into a rectangular grid on the screen. There, they expand and can be moved around with a mouse or rotated 90 degrees with a double click. Once a piece is placed with correct rotation in the correct position (with some leeway for small errors), it snaps in place and cannot be moved anymore.

Features.

Design.

The overall design is defined in CSS, while it is quite detailed, it mostly uses basic concepts seen in tutorials. Some more notable aspects of the CSS design are customization of elements such as scrollbars (toolbar and information panels visible in all games), text, and number inputs (*scream* and *map*), speech bubbles (*scream, labyrinth*), or tooltips (hints in the toolbar and ciphered coordinates in *map*), and a lot of keyframe animation used for smooth transitions and user experience. Additionally, many design aspects were utilized to help the user navigate at times unconventional interfaces (for example, changing the cursor appearance for elements that change behavior (sometimes clickable, focusable, draggable, or none of the above), for example in *dots, scream, film,* or *puzzle.* Lastly, certain games' design was created using almost exclusively canvas elements, especially for complex animations (*dots, blackout, bat, labyrinth*).

Canvas-Based Attributes.

Apart from using canvas to animate characters (*dots, scream, bat*) or elements (*film*), complex canvas animation was used primarily in *labyrinth* for a 2D maze, *blackout* (with a sound-reactive gradient), and *bat*, where the bat's movement was animated on a canvas with 'gravitational' calculations (to make the movement more realistic). Furthermore, the detection of collisions was tailored to each sprite of the flying bat, performing calculations based on the position and shapes of the bat and the tubes (graves). This was done using geometric 'collision masks' made for each sprite separately. The masks were purposefully made simpler and slightly smaller than the bat to make the game more lenient when it comes to almost unnoticeable collisions.

User Interaction.

Apart from 'click' and 'keydown' events, certain games heavily used 'drag' and 'drop' – *puzzle*, where the elements are moved from the toolbar to the puzzle grid, where they can be moved around and rotated; 'mousemove'/'mousedown' events in *dots* (practically using the dots on a canvas as buttons); and 'input'/'focusin'/'focusout' in *scream* and *map*. In the former, input boxes are used for each part of the given phrase separated by a symbol or a space. The use of 'focusin'/'focusout' events allows for uninterrupted typing – automatically moving to the next box once the needed number of characters is typed, as well as seamlessly going back to the previous box with a backspace to change the word entirely or correct a typo.

Randomness.

The majority of the games have some randomness to them. In some, element(s) are randomly selected for the game from a specified list of possibilities.

Such as the 12 movie scenes chosen from 24 possible ones in *film*; the phrase to guess chosen in *scream*; or the dot connections in *dots*. Additionally, the progression of certain games uses randomness, such as the spinning wheel and actions of unplayable characters in *scream*; the randomly generated 2D maze in *labyrinth*; the generation of columns (graves) and background clouds in *bat* as the game goes on; and the randomness of the alphabetic cipher used in *map* (ciphering both single- and double-digit numbers to make the game less repetitive).

Time.

The project uses a lot of asynchronous functions, time delays, and promises. This is often done for animations (not interrupting them) or creating a more immersive experience. With delays, unplayable characters participating in the game may seem more real, as they do not act instantaneously (for example, in *scream*). Additionally, this allows for better character dialogue with pauses (in *scream* or *labyrinth*). Moreover, interesting effects can be created thanks to time delays and asynchronicity, such as the 'typing' effect in the introduction of the whole project.

videos of each game can be found here:

https://drive.google.com/drive/u/1/folders/1V6kQlMCvMJa2pp0oLdorW7Ip-WJp1G4c