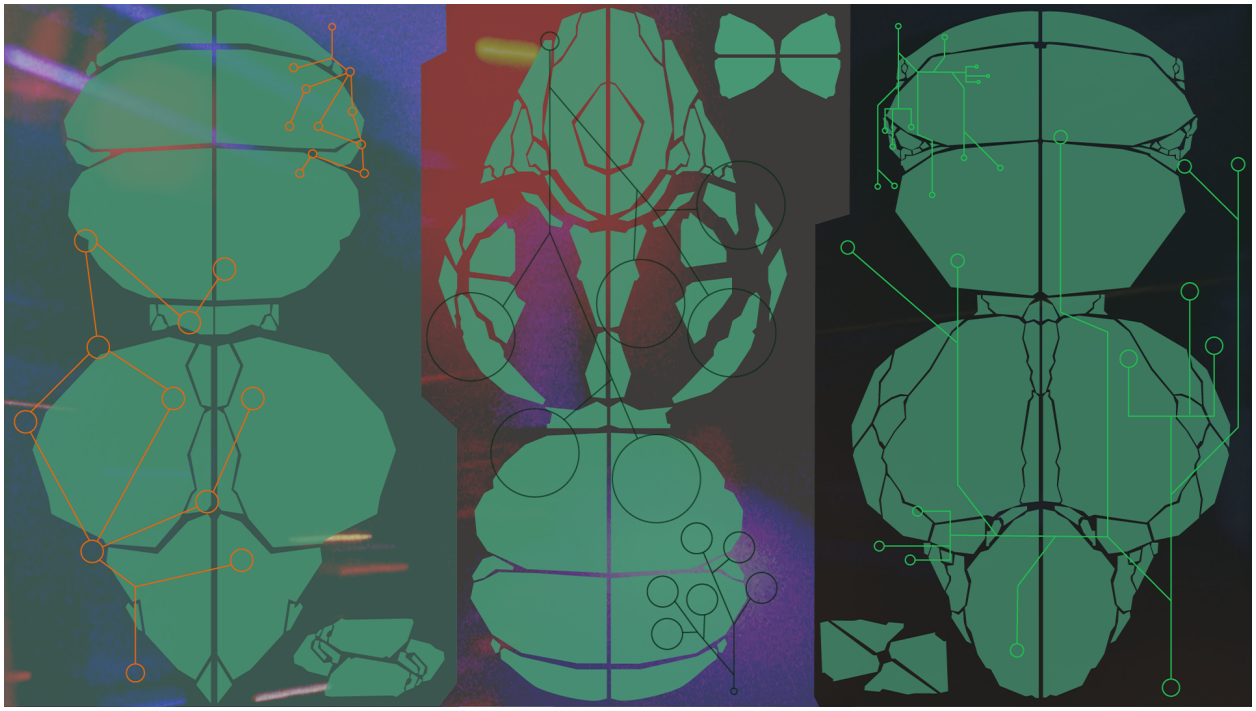
Crab Simulator



(Figure 1.0)

Crab simulator is an audio visual experience consisting of a modular crab creation menu flow and a simulation ‘test run’ stage of the user selected crab, with the ability to either succeed at the navigation of a terrain/maze, restart and  create a new crab from scratch, or try to preserve the simulation indefinitely by consuming corrupted q-bits that threaten it’s destruction.

The modular crab creation menu flow consists of the following stages:

- Crab Template selection: Select one of three crab templates to start with. Crab templates are laid out vertically in three sections of the canvas (see Figure 1.0).
- Crab editing interface: Consists of a single crab template laid out horizontally across the canvas. The head, the carapace and the abdomen, as well as the set of limbs represented by a mystery ‘limb pack’ can be changed by cycling through available options.
- Confirm crab selection screen: A message asks the user to confirm their selection and transitions into gameplay.

Simulation stage:
A pre-loaded 3D model of a crab is displayed on screen with a fixed camera 3/4 top-down orthographic perspective view. The crab’s parts correspond to the user’s selection in the crab construction menu.

Possible user interactions:

- Crab can be displaced around the scene using the WASD/ Arrow keys. Avoid fixed obstacles and reach the end of an isometric maze.

- Crab can be displaced along a terrain and individual limbs rotated by user input with an overly complicated control scheme.
- Limb movement is mostly inconsequential to the crab construc's ability to navigate it's simulated environment. It does, however, modify it in the following ways:
    - limb movement increases crab acceleration.
    - Pincer engagement decreases crab acceleration, and turns consumable objects (and small obstacles?) into mush on contact. Mush can be consumed as normal.
- Crab can consume or destroy objects in its wake, placed in the environment at set locations and representing people, human-made things like cars and schools and houses, civilization, and the very fabric of reality.
- Quantum black cubes rotating on their axis at large speeds (corrupted q-bits) can be generated and placed at random positions in the scene over time, as a by-product of crab simulation. If the amount of corrupted q-bits rises above a tolerance threshold the simulation ends.
- Crab can consume corrupted q-bits to preserve the integrity of the simulation for as long as possible.

Audio and visual effects:

- Objects and q-bits can change positions by slight amounts as the corrupted q-bits reach critical levels.
- A transparent overlay of the created crab construct rotating in space can be displayed on command on top of the simulation for user inspection.
- Menu interactions are accompanied by 'glitchy' sound effects.
- Gameplay features SFX for consumable items, crab movement, q-bit corruption and maze completion conditions.
- Other 2D transparent overlays (crab template schemas and textural images) may appear on screen at set intervals throughout the simulation.
- The mouse cursor is replaced by a crab pincer icon.

Notes:

WASD movement in 3d coordinate space:

- If statements to deal with negative x, y values?
- Omit movement along z obviously.
- Move the camera (or environment) to be completely within the positive quadrant (2) of the xy plane and restrict movement to that?

Collision with obstacles:

- Only boxes of varying sizes (large, medium, small) as obstacles and maze End point.
- Check for xy overlap with the base of cubes? Basically square with square overlap then subtract from crab x,y until it no longer overlaps.
- Tile based movement?

Maze generation:

- How? Random placement? Manual fixed placement?

- Tile based placement?
- A few generated layouts with interchangeable parts?
- Control distance of end point to crab?
- Separating layout into components, randomly place those?