Final Project Proposal

Alex Ceberio & Iñaki Soler

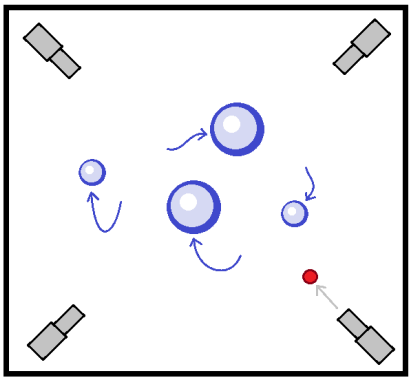
Proposal 1: **Smart Bubbles**

- Main idea:

Our idea is to create a multithreaded AI of bubbles that can interact between each other and dodge bullets that are shot randomly aiming at them.

- Demo and AI behavior:

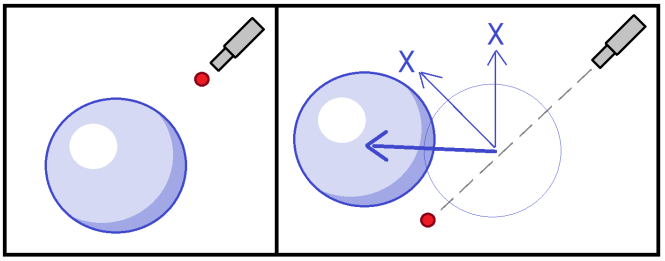
Smart Bubbles Demo



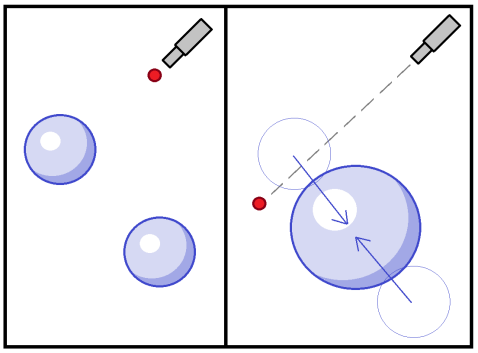
At the beginning of the demo three bubbles are randomly spawned. They move with a random direction through the screen with a fixed velocity (the movement will require physics and collisions). Four guns will also be spawned on the corners. They will be aiming at the bubbles and shooting bullets consecutively.

The moment of shooting is when the AI part comes. With a raycast system, each bubble will check if there is a possible collision with the bullet in the near future. In case there is, bubbles are going to try to dodge bullets with three possible methods:

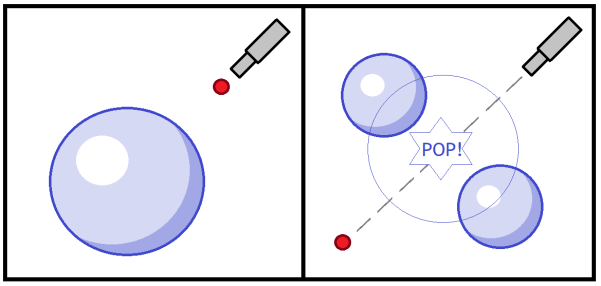
1. **Direction changing**: the bubble just checks all possible directions around it. If there is a direction where it can move and avoid the bullet, that's what it will do. Otherwise, another method is required.



2. **Joining others**: bubbles can split to dodge bullets (which is the third option), but they prefer to join another bubble than to split itself. If a bubble is small enough to join another without overpassing the maximum size, this method will be a possible option. For this it is necessary the close presence of another small bubble. If it can avoid a bullet this way, that's what it will do.



3. **Splitting itself**: As mentioned, a bubble can split itself. It's the easiest way to dodge a bullet, but that will imply creating more and more bubbles until they reach the minimum size, and that's not what we want. For that reason, this technique is the last option bubbles will take.



If none of those options are viable to prevent them from colliding against the bullet, the bubbles will have to accept their fate and die (I mean, pop).

- Logic:

The bubbles will be part of a multithreaded process, which means that each bubble will be a thread controlled by main. They can add more threads when splitting and join threads when joining. The main thread will be responsible to update each thread (bubble), and the bubble structure will communicate to the main thread any changes on threads, like joining, splitting or popping.

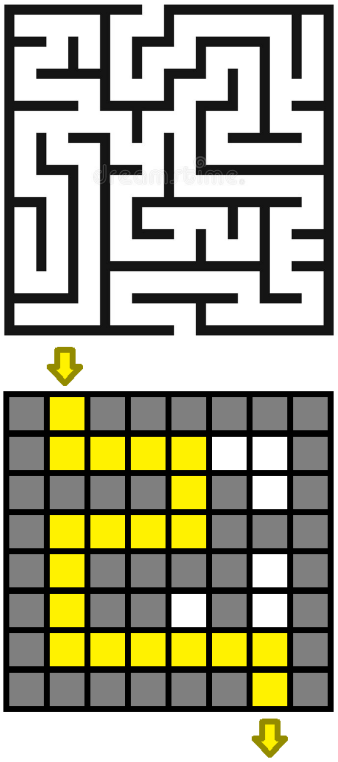
Proposal 2: **Maze Solver**

- Main idea:

Our idea is to create a maze solver using multiple threads to analyze the labyrinth and find the correct path.

- Demo and AI behavior:

The scene will consist entirely of a binary array whose values define either a wall box or a clear box. Each maze will be a scene to be loaded.



A maze-solver object will walk down the road analyzing all paths. Along the way different scenarios might arise:

- Whenever it finds a clear box on the sides, more threads will be created so that each thread contains and handles a different path.
- When the only clear box is the one behind, the thread will join to main and disappear.
- When a thread reaches the end point, all threads will stop working.

- Logic:

The process starts single threaded from the preset start point of the maze, which is an element inside the scene array, obviously a clear box. There is also a set end point clear box that the AI must find. The maze-solver struct will advance with a certain direction, which can change when it finds a wall in front. On each movement, the struct will check the boxes on the sides. For each clear box it might find, it will create a new thread and send it in the proper direction. At the end of the game all threads will join the main one.