Presentation

This folder contains a camera system I have created for the student game project between October 2016 and June 2017. It also contains the technical documentation that was made for the presentation at the end of the year.

The given code has not been modified since the end of the project to better reflect the work made. Also, these are only code snippets, so they may miss some classes to function correctly, like some important parts of the architecture which also belong to William Patruno, the other developper, and to a lesser extent Marc Bonté, the technical game designer of the project.

The cameraController folder contains most, if not all the system that is in charge of the game camera's movement. The camera moves with player characters (in code : Controllable, which contains an index corresponding to their joystick number), but also reacts to certain environment elements, like a landscape or a boss enemy. Also, it cannot move to a place where a player is outside the screen. One of its biggest advantage lies in the fact it adapts automatically while in game, which allowed game designers to test different settings on the go, without restarting.

Structure

The structure is quite simple : The cameraController moves the camera, and player characters or PointOfInterest tell it where to go. Its in its algorithm and mathematics it gets more complex.

First, the camera needs to find the barycenter of all players and where they are looking. This point, will be used to tell where the camera will try to point at. Then, we need to “clamp” this destination inside the camera's border, so that it can be reachable by the camera without making a player move outside the screen view. This is done in two steps :

1. First, the borders are defined by the camera's view minus a small portion of it. It is a trapezium and not a rectangle, since the camera view's rectangle is not parallel to the players' plane, so its projection is deformed. To put it in a picture, it is like the light of a flashlight on the floor : if the flashlight is looking perfectly perpendicular to the floor, the shape is a circle. But if it isn't, light shape changes to an ellipse.
2. Then, we check that each player is inside the limits. If not, we limit the camera's movement in the opposite direction of the player. For instance, if a player's is on the left screen border, we need to stop all movements to the right.

After setting the center the camera will be pointing at, we make the movement, by taking into consideration the distance of the camera it will take – based notably on the relative distance between the players -, and the rotation of the camera. It is done by using mainly trigonometry and some speeds calculations to ensure a smooth movement.

On the PointOfInterest's side, they are added to the camera when something triggers them (usually when a player enters its influence zone). When activated or deactivated, its strength is changed dynamically so that the camera's transition is as smooth as possible, especially when they are switching state quickly.