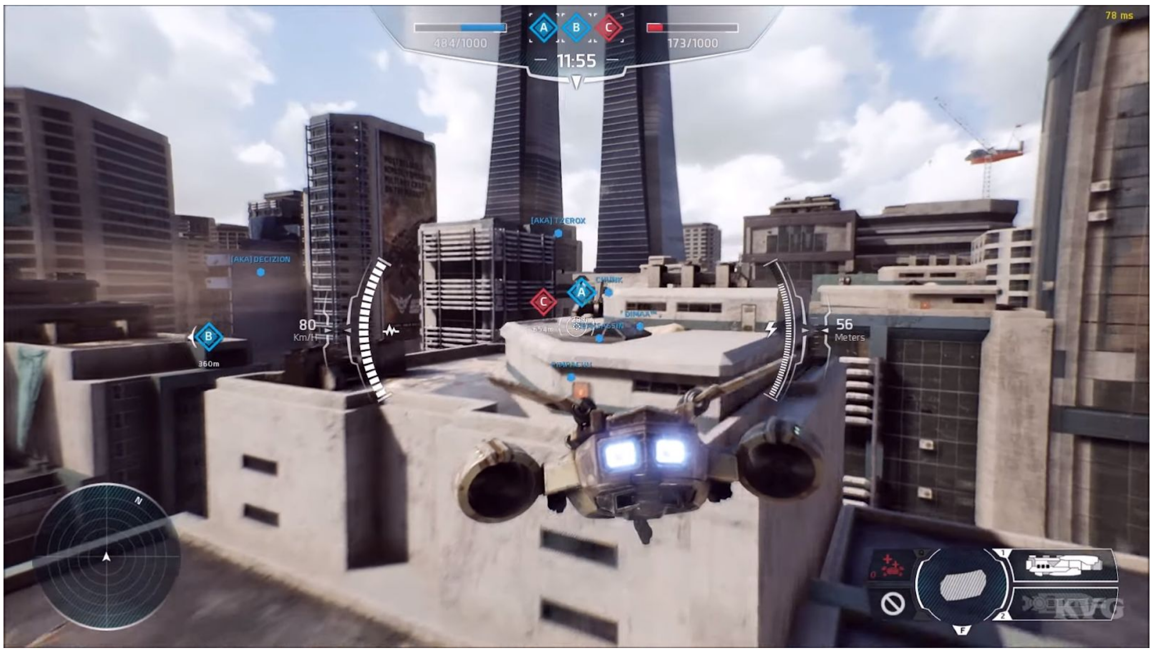
ANOTHEREALITY

Technical Assignment: Drone fight gameplay

The game

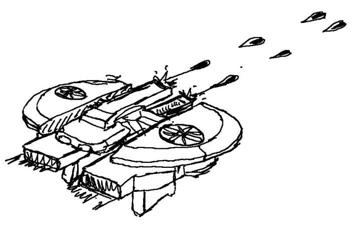
Drone fight is a fighting action game where the player pilots a drone in third person and has to destroy various turrets placed in a city-like scenario.



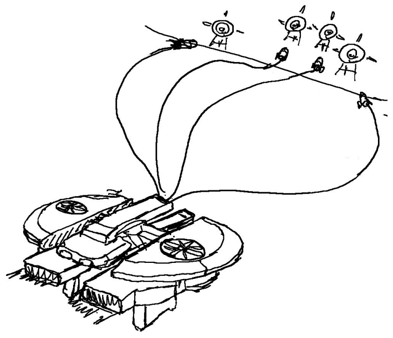
More in details, the player controls the drone using keyboard + mouse:

- The drone uses **WASD to move horizontally**, parallel to the ground, ignoring the camera vertical orientation. Double-press one of the keys to do a quick dash in that direction!
- Q/E (or CTRL / Spacebar) moves the drone up/down vertically, perpendicular to the ground
- Mouse controls the camera, used to aim for shooting and determining forward direction.

The drone has two weapons:

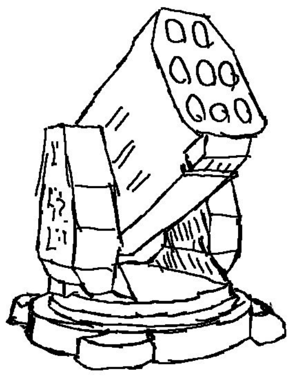


Main weapon (left mouse click): shoots 2 **bullets** repeatedly with infinite ammo. Bullets fly straight towards aim, no ballistic.



Sub weapon (right mouse click): keep pressed to acquire targets (instant upon aim), release to shoot **homing missiles**. Cooldown between barrages (*initial value: 3s*)

Turrets are stationary enemies that will shoot a barrage of 8 missiles at the player if he enters in their detection range and is in line of sight.



Turrets have a lockdown-time once the player gets in sight (*initial value: 2s*), and cooldown between barrages (*initial value: 5s*)

Note: you can just make the turrets look at you, no need to handle rotational axis animation like the sample image.

In the level there are 50 turrets. Player and turrets have a maximum life. If the player dies to the turrets, game over and restart. If the player kills all the turrets, win!

As a player, you also have a special skill: Time Rewind.

Time Rewind has a 20 seconds cooldown, and it will rewind everything to 3 seconds before, except for the player drone position and projectiles. So it will rewind:

- Enemy homing missiles
- Player's HP
- Turrets state (for example, if a turret is destroyed but was alive 3 seconds before, it will be restored).

Rewind effect should be visible (you see the homing missiles go back in time, not just a teleport), it has no range limit and the player can continue to move while rewind is taking effect, but without being able to shoot.

Your tasks

Your objective is to **develop a videogame** (PC Windows support), **where you can play Drone fight,** as it is described above.

For the development keep also in mind:

- Pay attention to the third person flight experience. Turning around with the camera shouldn't turn directly the drone, but the drone should catch up with the camera with his maximum turn speed.
- Balance homing missiles "precision" and movement countermeasures (dash or movement inversion).
- Develop a **functional** UI that provides the feedback needed for the gameplay mechanics.
- Graphically speaking, there are no requirements on UI / 3D / Artstyle (you can use basic shapes), but the game should be easy to read and understand.
- You're free to design the environment and place the 50 turrets as you want.
- The rewind effect should be "continuous" while it is in effect: you should clearly see elements gradually going back in time.
- **No external plugins for gameplay features are allowed**. The game architecture should be designed and made by you alone. You can use external plugins and assets for graphics and audio.

For the game visual aspect 3D is required.

For any information missing, you can make your assumptions and make decisions autonomously. Keep in mind that the expected result is a game that's **playable and enjoyable**!

Development mode & Tools

You're free to use graphical and audio assets downloaded from the internet. **The project has to be developed using Unreal Engine 4 and C++ language or Unity and C# language**, you're free to choose!

Blueprints are allowed but greatly discouraged, use them for graphical and superficial functions, nothing from the base and architecture of the game should be on Blueprint.

All the code produced has to be your own, without any use of plugins.

Project should be developed using a repository system (Git, Subversion, Mercurial, Perforce...). No restrictions about chosen hosting.

Deliver the results of the test both as an **executable application and with the source project**, along with a simple document that outlines the technical architecture adopted (main scripts, components and how they're connected).