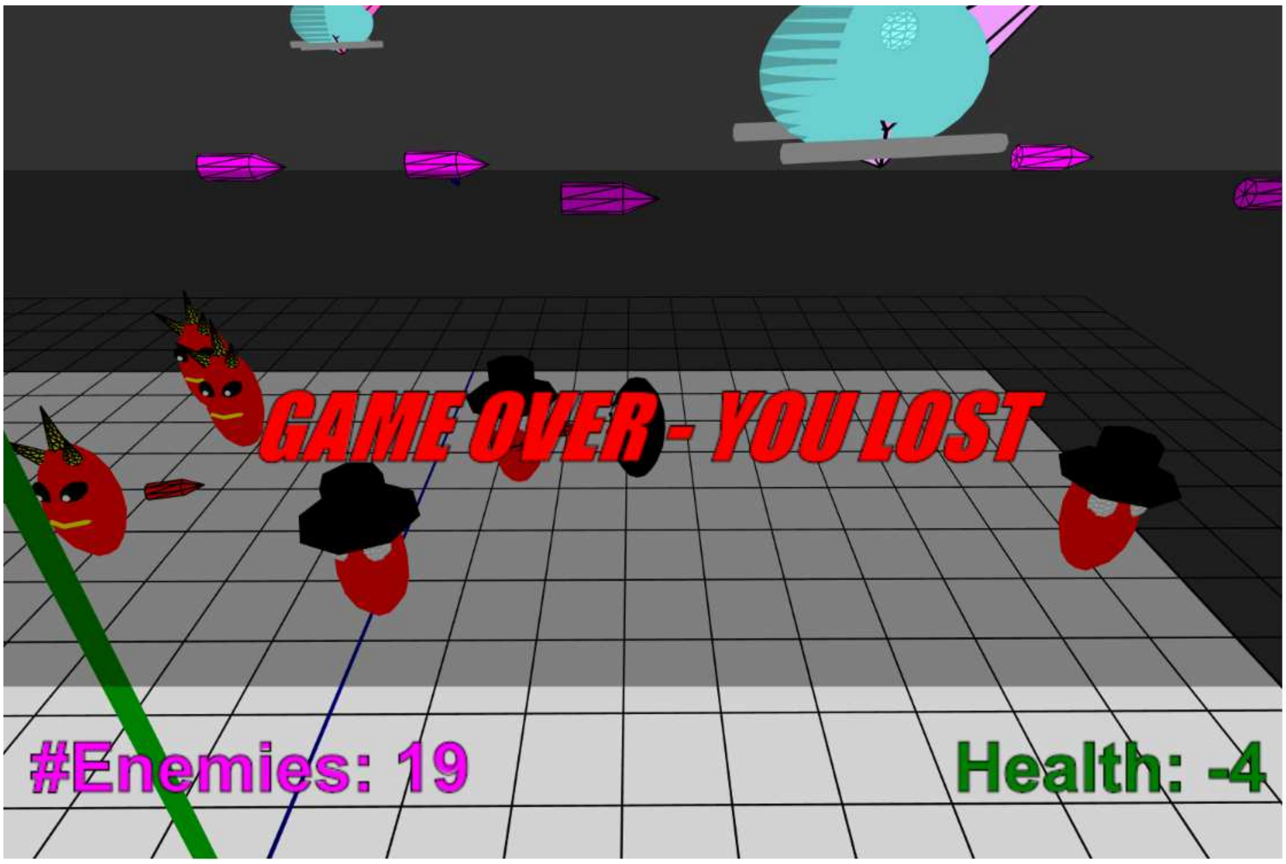
731 Final project (P5.JS) – version1

Interactive animation in 3D (P5.JS)

1. Description

In this project, you have to create **an interactive animation in 3D** (optionally a videogame, including some in-game feedback and and end-game result, winning or losing).



A straightforward approach for this project is **to start from the official starting point published by the teacher**, that is mostly based on the latest activities done in the classroom (the additional features included for the convenience of the students are explained below).

In case of popular demand, the teacher may publish in the future some additional technical papers to help development of the projects.

1.1 Formal requirements and ordinary extras

There are a number of **formal requirements** of the project:

1) Implemented in Javascript/P5.JS, using the 3D/WebGL engine.
2) Canvas resolution: 750 x 500. It is possible to add one or two supplementary 2D canvases on the sides in order to display additional in-game information, but since the technique for overlay canvas (2D) is complete, it is probably a better solution to display such additional information directly on the main canvas (overlay 2D canvas).
3) Controlling a protagonist object (the "mySelf" object) with the keyboard.
4) Keeping information of at least one enemy (ideally multiple), that have an autonomous behaviour,

The starting point consists in a 3D world in which the user controls an object on a horizontal plane (with the arrow keys), and there are a number of enemies. We can shoot at them, and they can shoot at us. **Note that the starting point already complies with all formal requirements.**

The project can be radically different, e.g. a kind of PONG game, where we need to move a slider and score goals with a bouncing object between the automated enemy's position and us. Talk to the teacher if you want to pursue something totally different, because your idea may depart too much from the contents of the course and it may not me realistic to complete within the available time.

The project description is "open" on purpose. You are free to focus on those aspects that you are most interested in (e.g. aesthetics, or mathematics/physics of objects, creative new shapes of objects, narrative linking the existing elements, etc).

In case you wish to focus on aspects barely covered during the course for technical limitations of our hardware (e.g. lights, light sources, textures, music) please consult the teacher.

Expected / ordinary features in the project (not mandatory to do all of them, at all).

1) Create new types of enemies:
   a. New appearances.
   b. Other types of behaviours, e.g. chasing the protagonist.
2) The same applies to projectiles shot by the protagonist and the enemies. Currently projectiles have a linear movement. Ideas for different types of movements:
   a. Parabolic trajectories, including ascending movement at first, and over time downwards acceleration (until object touches ground or Y coordinate negative).
   b. Chasing projectiles, that will always advance towards a moving target.
   c. Other exotic trajectories or behaviours (e.g. splitting projectiles, orbiting objects that protect us against enemy projectiles, etc).
3) Some structure in the health levels of protagonist, enemies and projectiles, damaging capacity of projectiles or enemies with direct contact, etc.
4) Other types of information displayed to the user (auxiliary side-canvases or preferably on the 2D overlay canvas of the starting point).

1.2. Extraordinary features in the project

Other ideas that would involve more work and would be more highly valued (because of their inherent complexity):

1) Other elements (trees, blocks, walls) that may have a mere decorative purpose or interact with the mobile objects. Such interactions could be:
    a. Prevent protagonist or enemies to keep advancing in that direction (e.g. trees or walls).
    b. Discounting health points or killing moving objects (enemies, projectiles, protagonist) e.g. chainsaws, fires or other traps.
2) Additional movements of the protagonist (or the enemies), for example:
    a. Moving sideways (currently only rotates on its own axis and moves forwards/backwards, both isolated or combined).
    b. Jumping.
    c. Ability to navigate also upwards/downwards (e.g. like a flying airplane, helicopter or dragon).
3) Objects having intrinsic changes in their appearance (e.g. moving elements like articulations, legs, wings or rotors), changes in color over time or according to health.
    a. A particular case would be a special case when the object is changing state, e.g. when dying not disappearing immediately, but having some special change of appearance like ascending, or rotating, changing color, etc.
4) Sophisticated / pseudo-intelligent behaviour of enemies, for example becoming more aggressive or faster when the protagonist is approaching them.
5) Interaction of protagonist with certain objects, for example special key to allow the protagonist to start using a car or motorbike, making it much faster, resistant or with more firepower (e.g. as in grad theft auto or other videogames).
6) Etc, etc. If you're not sure about how you could implement an idea, ask the teacher to guide you (or to inform you that the idea is too complex for the time available).

.

2. Evaluation

2.1. Deliverables

In total, every student should submit 4 files:

1. A JAVASCRIPT file (`731surnameofpupil.js`) that contains the project. The student should include his/her name in the comments section on top of the file.
2. An HTML file (`731surnameofpupil.html`) that allows the execution of the project.
3. **At least** one PNG image (`731surnameofpupil1.png`), that will contain a **screenshot showing the appearance of the project.** Alternatively, **the student can attach more than one image if this makes the demonstration easier** (e.g. one image for the first object, another image for the second object).

  a. The student may choose to put **more than 1 image**, numbering accordingly such additional screenshots.
  b. Make sure that of both scalable objects there are several examples, some smaller and bigger, and that for the more customizable object there also
  c. In Windows, a simple way to take screenshots is using the combination Alt+PrintScreen with the browser as the active window, and then capture/paste the image from memory into a typical image edition software, like Microsoft Paint.
  d. If you keep the overlay 2D canvas on top of the 3D canvas of P5.JS, the PrintScreen method is the only one possible (RightClick→Save Image won't work, because it will capture an image of the 2D canvas, but not of the whole thing).

4. A TeXT file (`731surnameofpupil.txt`) as a **self-assessment**, where the student will explain, in his/her own words, the elements/merits of the project (good to refer to the list of suggestions/ideas provided in this document, about features that would be valued).
   a. **The name of the student** should be included also in the beginning of this this TXT file. The student will explain all the "merits" of the project.
   b. **An expected grade** (ranging from 0 to 10) should be indicated.
5. **Any other "asset" used by your project**, such as JPG or PNG images for the 2D canvases, JPG/PNG for 3D textures (formatting 3D textures with base64 into a JS file is preferred), MP3 or MIDI files for sound (if any), etc.

Given the current situation and risk of distance teaching, the project may be submitted using any of the following methods:

a) **Teams / Assignments**. Here you can upload the deliverables separately or packaged inside a ZIP file (zip file is preferred).
b) **Teams / Private chat**. The same, you can transfer the deliverables to the teacher separately or packaged inside a ZIP file.
c) **Email (Outlook)** to the teacher. Here you have to send the files packaged inside a ZIP file, because **email servers block files with the JS extension**.
d) **The special folder 999submit in your Y: Drive.** This option is only possible during classtime,

2.2. Deadline

There is an "**intermediate deadline**" on **Monday 19 April 2021 at 23:59**. The purpose of this deadline is to ensure that every student has the project ongoing (at least started), and there are no last-minute surprises, given the importance of this project.

The **final deadline** is **Friday, 30 April 2021 at 23:59**. Since the technical interviews will take place on the week afterwards, later submission might be accepted but the technical interview would take place even if the project is not finished yet.

2.3 Evaluation

The evaluation of this project leads to the B-mark of the second semester in S7ICT, therefore it is as important as a final test in other subjects.

A project not submitted leads automatically to a mark of zero.

Note that a plagiarized project (e.g. checked during the technical interview) totally or in part may imply a substantial discount in the evaluation.

The evaluation process has 3 parts:

1) Assessment by the teacher of the deliverables.
2) Continuous monitoring and Technical interview.
3) Optionally, presentation of the project in class.

Assessment by the teacher of the deliverables (mainly the source code and the execution of the project). This is by far the biggest aspect (in normal cases). Aspects considered:

1) Brute work, i.e. brute amount of work (in brute terms). Brute work is objectively measurable by counting lines of code, number of functions / classes created, number of shapes making up complex objects, etc.
2) Sophistication of the project, mainly related to inventive aspects like special trajectories or behaviours of the enemies, and other extras mentioned above. This is generally more highly valued than brute work as such.
3) Aesthetics.
4) Originality.
5) Quality of the source code (avoid that it is too messy).

The technical interview can take place either during development before the deadline, or in the weeks after the deadline. A technical interview provides a clear picture about the level of understanding of the project by the student (in extreme cases detection of plagiarism), and it also helps the pupil to be aware of this evaluation technique so much used in later studies like universities or technical schools.

The technical interview consists in a conversation of 5-10 (or a bit longer) minutes with the teacher. You may be asked simple questions (or even short exercises) that any developer of any project would be capable to answer easily, such as:

- Show me the code / lines responsible for rendering this object here.
- Show me the code / lines responsible for this special trajectory / behaviour.
- Change the color of this enemy here.
- Make the wheels of that car bigger.

The projects progress will be monitored regularly. Students who tend to work only at home are expressly required to bring to the school (USB stick or other means) their latest version of the project, so that the teacher can see the evolution. A project never seen before by the teacher will be subject to deeper scrutiny during the technical interview.

The presentation (fully optional, for pupils who want to practice sales skills) consists in a short presentation (not necessarily powerpoint, it can just be the project being executed), in which you describe publicly the technical features / merits of the project, what was the most interesting part, which features you would have liked to sophisticate more, etc, etc.

3. Further notes

3.1. About back-up (security) copies

Whenever you do a project, it is strongly advisable to keep back-up copies of it as you advance and add more and more features/complexity. This has 2 purposes:

- Avoid loss of precious hours of work.
- Justify that the work is yours (being able to show partial versions of the project).

Keeping copies in different media is highly advisable (computers of the school, a USB-stick, computer at home, as attachment (zipped) in an email message to yourself, printing your code occasionally, etc). **Any time you complete an element of your project (or any element), and is being drawn properly (no errors), may be a good moment to do a back-up copy**. You are also welcome to send to the teacher a partial (incomplete) version of your project at any time, to minimize the risk of getting a zero for failure to submit the project in the end.

3.2. Contact / Further help from the teacher

During the next weeks until the deadline, you will be able to work on this long project during classtime. Further information or technical aspects may be explained and/or distributed as printouts, depending on demand.

The best way to start this project is **the official starting point for this project 731 provided in Teams and SMS**,
Under demand, other materials might be published in the Files subfolders in Teams.

You can also contact the teacher by email. Email address of the teacher:

hernanje@teacher.eursc.eu

In case of any doubt or question about the task description, requirements, please ask the teacher

ANNEX 1: The starting point

The starting point is a rather large JS file (also an HTML file) comprising roughly 2000 lines of code (mostly 3D composite models made in previous activities).

In this annex it is provided a kind of "map", to facilitate navigation during the development, especially in the beginning.

Note that as you insert new code into the project, all these line numbers will change (a lot).

Lines	**Part of the project (functions, classes, purpose, etc).**
1-30	myRandom and other utilities (for future inclusion, e.g. conversion HSL to RGB, etc).
30-255	Subsystem for Canvas 2D classes, used to facilitate depiction of overlay messages in the auxiliary canvases. Most of the code here is "private", i.e. it consists in auxiliary functions used to draw/write elements in a standard 2D canvas in javascript (Canvas API, seen in S5). The important functions are the refresh1(), refresh2() and refresh3() Classes: MamerExtraCanvas2D MamerExtraOverlay: sub-class, it could be assimilated into the ordinary canvas 2D, and possibly much better solution than the other 2.
255-595	Subsystem for Teacher's orbit control. All camera modes (keys 1, 2, 3, 4) are controlled and processed here, and also the user controls for moving the protagonist (currently arrow keys and "A" key (code 65) for shooting. Variables (267), including cameraMode, initial angles, etc. myCheckingKeys (393), e.g. for player interaction. teacherOrbitControl (502): for handling the camera modes.
598-660	Mamer Protagonist class, includes most of the special actions of the protagonist.
660-1080	MamerObject (660): Parent/common class for all enemies, projectiles and also other passive/decorative objects. MamerBall (742) MamerBallLight (750) MamerHomer (792) MamerRocket (810) MamerHelicopter (826) MamerBullet1 (850)

	MamerEnemy1 – with hat (911) MamerEnemy2 – with horns (953) MamerEnemyBullet1 (1035)
1087-1646	Basic 3D elements from early activities and other 3D composite elements. myAxes (1087) myPinetree (1129) mySimpson (a.k.a. Homer): Line 1167 myHomerOriented(1228) paintRocket2oriented (1263) , paintRocket2 (1290), paintRocket1 (1300) paintHelicopterOriented (1326), paintHelicopter1 (1356) myZombi (1441) myZombiOriented (1493) myHomerHell (1523) myHomerHellOriented (1616)
1667	mySceneMovements() This function is in charge of updating movements and other changes in objects, also cleaning up the global arrays (enemies, decorative elements, etc) when elements die.
1737	myScene() This function is called every frame, and positions all elements in the 3D scene.
1841	Declaration of important global variables, such as myself (the protagonist), global lists of elements, and others. This includes the subsystem to spare computing cycles when the browser is not on focus (user opening another window), contribution of pupil Philip Furche.
1878	Setup() This function initializes the whole application (standard name in P5.JS). If at some point it’s necessary to load external assets like textures, that should be done in the “preload() function”, normally placed just above setup().
2005	Draw() The true core of any p5.js application, executed once per frame.