# Project planning

## Project scoring regulation

In this document we will have the phasing of the project with details of tasks and penalties.

**Observation:**the scale is extensible, as some students may want to implement additional things not specified in the scale. in this case they must message me before implementing them so that we can discuss a statement for the scale and add that task for others who want to implement it. This fact affects the grade only positively, because even without these tasks you can get 10 - such a task only brings extra options.

### Bonuses and penalties

**In the week with a normal score, it can still be with a bonus in the following situations:**

1. the student presented during bonus week but had mistakes or incomplete assignments and decided to correct the situation
2. I rescheduled someone for justified reasons: technical problems (one of us didn't have a computer or internet, etc.), administrative problems (he couldn't present because he had problems with the dormitory, the secretariat, etc.), health problems, he is a serious person and he would have completely finished the topic but he got busy with other subjects, etc.
3. an official postponement, mutually agreed upon by all students, additional explanations being required in class/laboratory.

**Careful! The normal scoring period will not have a bonus applied if the student presents for the first time with the project stage resolved and without a justified postponement.**
**bonus**it is worth 10%. 15%
**Penalties are 10%.**

### Other details

- **What matters is the time you sent the stage and not the time of presentation.**You can send the stage even if it is somewhere around 90% ready or with small bugs and I still consider the bonus as long as you tell me the situation, and you can even fix the problem before the presentation. (obviously the bonus is not considered for a stage that has just begun or with a lot of wrong things). I also consider the bonus if you are a few minutes late (not a few days!). So my attempt is not to see the deadline as something stressful, but on the contrary to have the courage to send what you have worked on and ask questions about what did not work.
- **A stage can be presented at any time during the semester**but it is good to present as soon as possible when you have the code fresh in your mind. Also, if you have mistakes or missing things at a stage, which would lead to de-pointing, you can recover the points in the next stage, if you modified/completed those things within the project.
- **In the case of de-points resulting from mistakes or missing tasks, the score can be recovered if you fix those problems by the next presentations.**Points are not lost forever, you can make additions in week 14 for stage 1 without affecting the stage bonus or penalty already written in the table.

- **Please send the stages to teams**(html file), not by email or on the website. Those who have put the project on github just send me the link or if they have already sent it, notify teams that they have made the change (the message with the notice must be sent before the deadline, so I can check).
- Stages from 1 onwards must be presented to get points on them.
- **Bonuses within a stage can also be resolved later without penalty**(i.e. you can, for example, do the bonus after 3 weeks from the deadline) but the penalty or time bonus of the stage will be applied to the bonus. For example, the stage was done with a time bonus, and after 3 weeks from the deadline you also send a bonus task, the time bonus of the stage will also be applied to him (and if there is a time penalty, the penalty will be applied to him). The reason: encouraging the student to do the obligatory tasks on time.

## Useful links for the project

https://www.pexels.com/search/texture/free textures

**Observation.**The recommended score is a minimum required to have a project with a recommended grade (= 6p for full-time education). The maximum score next to each category is the sum of the maximum scores for each sub-item (if you did the maximum required elements, with maximum complexity).
The term "recommended" does not necessarily mean to stop at that point, it is only indicative (for example for creativity I put recommended:0 because it is not mandatory to come to the project with something unprecedented, but it is still encouraged to have original ideas, etc.)

| **Stage** | **Bonus period** | **Normal scoring period** | **Since the weekly penalties start** |
|---|---|---|---|
| **Stage 0 (recommended score 0.5) - choice of theme and description** | - | has no bonus;<br>it is sent until 14.10.2024 because if you don't have the project theme, you have no way to work on the next stages | |
| **Tasks stage 0 (0.5))**<br>1. (0.02) Choosing a theme (finding a title for the game)<br>2. Brief description of the theme (google docs, 1/2-1 page). It will contain the following information:<br>a. (0.13) a general description of the game (0.13 divided into) :<br>i. (0.05) Context (the setting, the story of the game; what it's about)<br>ii. (0.03) In which category it falls (strategy, shooter, adventure, fighting, etc.). They can be combined categories.<br>iii. (0.05) The rules of the game: what are the possible actions of the player. In what situations does the game end? How to decide whether he won or lost, how to calculate the eventual score.<br>b. (0.1) Establishing User Interaction. It can be given as a description in words or a UML diagram (only for the main actions, not the whole game). This should include usage scenarios. What is the first interface the user sees. What actions are available when entering the app, etc.<br>c. (0.02) Identification of the user category (age, personality, interests, social setting).Example: the game is dedicated to vegetarians of all ages, with a medium to high level of general culture, passionate about cooking, strategy and ecology <br>d. (0.03) Establish keywords/phrases (a list of at least 5-6 keywords that describe the game). Reason: some game publishing platforms require such key phrases (keywords, tags, etc.)<br>e. (0.2) Searching for similar games (4-5 games) as a theme to note the pros and cons (minimum 2 positive aspects and minimum 2 negative aspects for each game. Games can be evaluated either based on a demo, watching some streaming - free videos showing the gameplay, and even with the help of reviews. No need to download / install the games. For each game you will write the name followed by the comments.For example if you want to make a village simulation game you will list simulation games for some kind of community (village, city, tribe, etc.) The games found do not have to be identical in idea to your game but just have some features similar. | | | |

| | | | |
|---|---|---|---|
| **Stage 1 (recommended score 0.7, maximum ...) - simple scene - terrain, materials, foliage** | 11.11.2024<br>-<br>25.11.2024 | 26.11.2024<br>-<br>the last day before winter vacation | the last 2 weeks of the semester |

Stage 1 (Static elements) Recommended score 0.7. Attention, the points are only taken on what is implemented by the student, not on elements imported from various packages!

Terrain requirements

- (0.05) There must be terrain in the scene. It is not mandatory to move on the field, it can serve as a landscape around the game platform.
- (0.15) The land must have a varied relief (there must be multiple low and high areas). The terrain will be assigned a material that includes multiple (minimum 3) textures (for example, grass, sand, rock texture, etc.). Associated textures should be painted on the terrain so that they are consistent with the shape of the terrain (eg a deep pit will have a rock texture and not grass/popcorn)
- (0.05) There must be at least one ramp on the terrain (Sculpt->ramp menu)
- (0.05) There must be two symmetrical areas on the terrain (eg two mountains). (see menu Sculpt->mirror)

Foliage requirements

- (0.1-0.5) Folliage mode will be used to add scenery to the scene. The student will explain how he used the folliage mode. The scoring is given according to the complexity of the settings, the decor, the different types of courts used, the student's ability to change the scene during the presentation.

Material requirements

- (0.05) Object with transparent material
- (0.05) Object with a metallic sheen that reflects its surroundings
- (0.1) Object with glossy material (which reflects the environment) on certain areas and non-glossy on others depending on a certain pattern (the solution will be done through blueprints)
- (0.05) Existence of an object with emissive color.
- (0.05) Object that has a 2-color gradient material (the gradient will be made by blueprints without using an external texture)
- (0.05) Object that has gradient transparency
- (0.15) Object that has a 3-color gradient material (the gradient will be made by blueprints without using an external texture)
- (0.1) Object that has a material that is transparent in portions and opaque in others. Portions of transparency are decided by a pattern (eg another texture)
- (0.1) combining the colors of two textures according to a pattern given by a third texture
- (0.2) simulating a color with sparkles (sparkly dots arranged randomly) using a noise node and without using an external texture (ie an image)
- (0.15) A complex material with a pattern created using at least 5 mathematical functions.
- (0.2) An animated material using the Time node, and at least 2 nodes with mathematical functions.
- (0.05) Using a normal map to create an object that feels like it has asperities even though it hasn't modified its vertices

| | | | |
|---|---|---|---|
| | | | |

**Pawn/Character; moving through the scene, player progress, recommended: 0.75, maximum: 2.35)**
A pawn and/or a character will be created for the user's movement
**Strictly pawn requirements**
(0.1) Realization of pawn by extending Pawn or DefaultPawn class

**Strict character requirements**
(0.1) Realization of character by extending the Character class
(0.2-0.5) Create character animation(s) to be used in game. The score is given according to how complex the animations are, the context in which they are used
(0.1) The class for the pawn/character in the scene is made in C++

(0.1-0.3) The character must have actions (Action Mappings) defined in inputs from Project Settings and implemented in the blueprint (examples of actions: jump, crouch, fly, etc.)
(0.1) Association of animations for the above actions (minimum one association)

**Common pawn/character requirements:**
(0.05) The pawn/character will have a (recording) camera added in components to follow the pawn in first person or third person style.
(0.15) Ability to switch from first person to third person tracking with the press of a key.
(0-0.4) Creating variables for pawn/character or other actors that reflect player state, certain properties (Each different type of data from those listed 0.05). Scoring is only given if the memorized information is relevant to saving the game:

- Integer (integer or Integer64)
- Boolean
- Rational (Float)
- Character string (String or Text)
- Vector (eg for storing a color, location, etc.)
- data table (Array) of any type
- set (Set) of any type
- map (Map) of any type

(0.1) The pawn/character must have the axis movements (Axis Mappings) defined in the Project Settings inputs.

- (0.1) It is added to the score if it can be translated on at least 2 axes defined as such
- (0.1) It is added to the score if it can rotate with respect to at least one axis
- (0.05) It is added to the score if at least one mapping is done for the mouse

(0.1) The pawn/character can change(increase/decrease) its movement speed
(0.1) The collision of the pawn/character with other objects will be dealt with using a collision box. The pawn/character will be able to pass through certain objects but not others (in at least one of these situations, one or more attributes of the pawn/character will change: for example, its health decreases if it touches an enemy)
(0.05-0.1) A scoring system. Depending on the in-game achievements a number will be calculated to show how well the player did.
(0.35) Highscore system. For final games, after the game is over, the score will be saved in a file, with all users' scores, ordered from best to worst (possibly the number of scores stored can be at most N, and any performance below the first N will not be saved). The user will have the option to see the scores.

**Damage system (recommended 0.25, maximum 0.25)**
(0.25) Unreal's default damage system will be implemented either on the pawn/character or on the actors the player interacts with. The ApplyDamage method will be used following an in-game event. With the help of an AnyDamage event the actor on which the destruction is applied will have some parameters affected. One case will be implemented for low value damage (object can change color, shrink, etc) and another for high value damage (eg object can disappear or change color differently from damage the small one, or to offer a written message on the screen).

| | | | |
|---|---|---|---|
| **Stage 3 - Trajectories. Projects (recommended: 0.5 maximum 1.75)** | | | - |

**Trajectories (recommended: 0.5, maximum 1.75)**
(0.2) Static trajectories. There will be in the scene the trajectories defined by spline curves created statically with the help of the editor. Actors will move along the route
(0.1) Speed that can be accelerated/decelerated along the entire trajectory
(0.25) Accelerated/decelerated speed only on certain portions of the trajectory (predetermined or calculated by the program)
(0.05) Stop moving on the trajectory
(0.05) Restarting the movement on the trajectory from the point where the object started
(0.05) Resuming the trajectory walk from the starting point
(0.05) Infinite walk on a trajectory.
(0.1) Completing the trajectory path a finite number of times (N) after which some in-game action occurs
(0.3 + bonus 0.1) Generate a spline curve dynamically, programmatically in the blueprint. If it is done via C++ 0.1 bonus points will be received
(0.2) Dynamically changing trajectory following an event.
(0.1) Existence of multiple objects on the same trajectory

(0.1) Dynamic addition (following an event or a certain game state) of additional objects on the trajectory.
(0.1) Changing the direction of travel on the trajectory, by program.

**Projects (recommended: 0, maximum 1.1)**
It is scored separately in addition to other score categories achieved by projectile implementation (that's why some scores seem small, because I considered that they add up to points on trajectory and collision events)
(0.2) Implementation of a special projectile actor. It will be launched on a trajectory following an event. The project will start from an actor (it can also be a pawn/character) with the aim of reaching certain coordinates. Projectile disappears when hitting a target (another actor)
(0.2-0.4) The projectile can track a moving target (adjusts its trajectory based on the target's coordinates). Scoring depends on the type of pursuit and the naturalness of the trajectory
(0.1-0.5) Application of projectile physics: projectile affected by gravity, wind effects, precipitation, different friction force in different environments

| | | | |
|---|---|---|---|
| **Stage 4 - advanced stage - the architecture of the game, of the level. Lights. (recommended 0.5; maximum 5.8p)** | | | - |

**Scene (map) architecture (recommended: 0.2, max: 3.4p)**
(0.1-0.5) is given for the complexity of the scene construction (the number of elements, the manner of placement, constructions created by placing elementary forms to obtain more complex forms). Using Foliage mode to create certain areas.
(0.1-0.5) It is given for the programmatic generation of actors with certain locations, rotations, sizes in order to create complex constructions (eg: a chess board made of cubes, a maze, a house made of wall-type objects and roof that have been placed through the blueprint to achieve the appearance of the house). The generation of actors in the scene will be done in the blueprint with methods like Spawn Actor from Class. At least one characteristic of the actors will be calculated by the blueprint (eg location, rotation)
**Game levels (maps):**
- (0.2) The game is multilevel with different maps. You move from one level to another following achievements in the game.
- (0.1-0.5 per level; max 2p for 4 levels) is given up to a maximum of 0.5 for each additional level, up to a maximum of 4 levels (the first level is scored in other scoring categories) depending on the complexity of its architecture (from point of view of terrain, skybox (or skysphere), lights, objects, placed on the map statically (with the help of the editor) or dynamically (through the program) skybox/skysphere, atmospheric elements, etc.
- (0.2) Using sublevels for map optimization

**Highlights (recommended: 0.2, max: 1.8p)**
(0.05-0.1) Relevant use of at least one directional light source (Directional Light). Modifying (static, manual) its properties.
(0.05-0.1) Relevant use of at least one point light source (Point Light). Modifying (static, manual) its properties.
(0.05-0.1) Relevant use of at least one spot light source (Spot Light). Modifying (static, manual) its properties.
(0.05-0.1) Relevant use of at least one rectangular light source (Rect Light). Modifying (static, manual) its properties.
(0.05-0.1) Relevant use of at least one atmospheric light source (Sky Light). Modifying (static, manual) its properties.
(0.05-0.1) Associating lights (in the form of components) of some actors (example: creating a flashlight, associating a Spot Light to the flashlight mesh)
(0.1-0.5) Changing characteristics of lights (such as color/intensity, being off/on) based on events/player state/time in game. It is scored according to the number of lights affected, the number of different types of changes and their complexity. Some examples (to give you an idea, but you can come up with something new):
- if the game simulates day/night sequence, SkyLight may vary
- Or we have some lights that are on for 5 seconds and off for 2 seconds, then on again and so on)
- light that turns on when we enter a room

(0.1-0.5) Animating the lights by changing the direction of the position, the dimming distance, gradually and continuously. It is scored according to the number of lights affected, the number of different types of animations and their complexity. Some examples (to give you an idea, but you can come up with something new):
- A point light source that walks on a path
- Simulation of a star whose brightness varies periodically over time gradually going from bright to dim and so on.
- A rotating spot source, like an automatic spotlight
- Directional light source that gradually changes direction

(0.05) Light source that does not cast shadows
(0.05) Object (actor) that does not cast shadows even though other objects lit by the same source cast shadows.
(0.1) Object unaffected by light (Unlit material)

**Random (recommended: 0.1 max: 0.6)**
Using random numbers in:

- (0.05) generating a random color applied to an object (actor, widget, etc.) in the game
- (0.05) random coordinates, rotations and/or dimensions for one or more objects or pawn/character
- (0.1) random string - for example for a password or part of the default username, or for saving the game
- (0.1) Random shuffling of the elements of a vector then used in the game
- (0.2-0.3) Probabilistically determined behaviors (0.2 is given for 2 complementary probabilities and 0.3 for more). Example: with a probability of 20% to generate elements of color c1, with a probability of 30% color c2 and the rest of color c3. Any element that depends on the probability can be chosen (color, location, shape, type of object, action performed, etc.)

| | | | |
|---|---|---|---|
| **Stage 5 Events. Collisions. Time manipulation (recommended 0.6 maximum: 2.5)** | | | - |

**By mouse (recommended 0.1 maximum: 0.3)**
(0.1) Relevant use of a click event within the game
(0.1) Relevant use of a begin cursor over event within the game
(0.1) Relevant use of an end cursor over event within the game

**Keyboard (recommended 0.1 maximum: 0.3)**
(0.1) Relevant use of a keydown event within the game
(0.1) Relevant use of a keyup event within the game
(0.1) Handling a key combination (between a special key - shift, ctrl, alt - and a displayable one, eg Shift+q, ctrl+w, etc.)

**Collisions (recommended 0.2 maximum: 0.6)**
(0.1) Relevant use of an overlap event within the game
(0.1) Relevant use of a hit event within the game
(0.1-0.4) Updating the data of the pawn/character and/or actor upon collision (hit/overlap) It is scored according to the complexity of handling the collision, for example, if an actor (it can be even the pawn) is in collision with different actors ( su different types of actors) to have different actions happen (for example, on collision with an energy bar, the bar disappears and the pawn gains health, but on collision with an enemy, the enemy just changes color and the pawn loses health.

**Time manipulation (recommended 0.2 max: 1.3)**
(0.1) Performing an action at a time interval t after an event has happened (For example, 2 seconds after the game starts, something happens), for example with a node of type "Set Timer by Function Name"
(0.1) Repetition of a function call at equal time intervals, with a node of type "Set Timer by Function Name"
(0.1) following an event or game state reached, a function called repetitively (at equal time intervals) with "Set Timer by Function Name", will be paused with "Pause Timer by Function Name" . Repetitive calling will be resumed after another event, with "Unpause Timer by Function Name"
(0.05) following an event or game state reached, a function repeatedly called (at equal time intervals) using "Set Timer by Function Name", will be canceled (repeated calls will be permanently stopped) with " Clear Timer by Function Name".
(0.2) Displaying the date (eg in a widget) using the now node and breaking the DateTime structure into components. The date will be displayed in day/month/year format (and if a number is below 10, it will be preceded by the digit 0)
(0.2) Display on the screen, during the game, the time that has elapsed since the start of the game or the beginning of the session, or since a certain event.
(0.3) For a time information (how many seconds until an event or how many seconds have passed since a time t, the time instead of being displayed as an integer number of seconds will be displayed in the format hh:mm:ss (h - hours, m - minutes, s - seconds). If any number in the 3 categories is below 10, it will be displayed preceded by a 0).
(0.25) Displaying in a widget the time of the last access or the time interval that has passed since the last access.

| **Stage 6 - Menu. Adding sounds (recommended: 2.2p max: 6.35)** | | | - |
|---|---|---|---|

## Game menu (recommended: 2p, max: 5.35)

(0.2p) Upon entering the game, the main game menu will be displayed. The game is not started (i.e. paused) until the menu is exited.
(0.05) Using an image in a panel
(0.05) Use of HorizontalBox and/or VerticalBox panels
(0.2p) Switching between menu screens using WidgetSwitcher
(0.2-0.4p) Creating a custom class for buttons. (it is scored according to how complex it is.

The main menu may contain the following buttons:

- (0.2p) New game start button. The button will have a suggestive text, for example "Start". Upon entering the application, the game is paused, and remains so until the user activates it
- (0.3p)The button to continue the last game started. When you click on this button, the game starts showing exactly the state in which it was left by the user before the last closing (or the last save)
- (0.2p) General settings screen (for player profile or new game features. The settings screen will be accessed via a button on the main menu. The settings screen will contain various inputs and a button to send data. Click on button settings will be saved in pawn/character properties.
- The inputsused in the settings screen (or other screens that ask for information from the user can be of the following types (caution, it is scored for each distinct type, not the input itself):
    - (0.05) EditableText
    - (0.05) TextBox
    - (0.1) Slider. Parameters such as: minimum and maximum values and step will be set.
    - (0.1) SpinBox. Parameters such as: minimum and maximum values, number of decimal digits, step (delta), growth exponent (step increase for larger values) will be set
    - (0.05) Checkbox
    - (0.1) ComboBox with at least 2 options
    - (0.1) RadialSlider Parameters such as: step, default value, etc.
    - (0.1) Extra 0.1 is given for ComboBox if options are added dynamically.
    - 0.1 is added separately for each type of input that is included in a custom widget in order to add new functionality to the completion (Example: a borderbox that changes its background color with each key press
- (0 - 0.3) Layout of the settings screen. Points are awarded based on:
    - Most important: Aligning elements (for example with a grid)
    - The fact that each input has an associated label (text next to it that says the role) and/or tool/type,
    - Change default colors
    - Readable text (chosen colors with good contrast; the text is not too transparent or overlapped with an image)
    - Using a nice background.
- (0.1) Adding elements dynamically (programmatically) to the widget, using methods like "Add Child to [container]"
- (0.1) Dynamic (programmatic) deletion of widget elements using "Remove Child" methods
- (0.4p) Old game load button. Clicking on this button will open a screen with the user's previous saves from which he can choose which game he wants. Saves can be listed via buttons or a combobox. The screen will be dynamically generated based on the files in your saved games folder. The identification of the games corresponding to the current player will be done by username.
- Show game information button:
    - (0.1) will take you to a screen with a text about the game explaining the story/context. The screen has a button to return to the main menu.
    - (0-0.4) Special styling of the game text screen. The score is given according to how complex and beautiful the stylization is:
        - using a scrollPane
        - using different colors within the text
        - using different styles: bold/italic
        - styling text as heading sections
        - using lists
        - the use of images within the text
- (0.1p) The exit button from the application (clicking on it closes the game)

(0.1) With the help of a widget, a menu displayed during the game will be created that will have the buttons (additionally dotted as follows):

- (0.1) Pause - when you click on it, the game pauses, and when you click on it again, it starts again. The button text should differ in the pause type, for example read "Resume"
- (0.1) A game exit button.
- (0.2) A button/key shortcut that pauses the game and brings up the main menu. in this situation the menu must have an additional button with the text "Continue".

(0.1-0.3) Displaying information related to the state of the player (or other actors) during the game. It is scored based on how complex the display is.
(0.1) Use a progress bar in displaying player information
(0.2) Option to hide and re-show the display during gameplay, for example when pressing a key.
(0.2) Simulation of some radio buttons using custom buttons
(0.3) Simulation of some radio buttons using custom checkboxes
(0.2-0.5) One or more informational screens that appear following an event or state the game reaches. For example, a screen where the player is informed that they have entered a new level or that they have "died". They are scored according to their number and the complexity of the display.
(0.1) Restart button in an info screen, in case the player died or the level ended
(0.2-0.4) Loading screen - is scored according to complexity

**Sounds (recommended: 0.2 maximum:0.5)**
(0.1) Added in-game sound
Additional points are given:
(0.1) The sound appeared within a widget following an event (eg button click)
(0.1) The sound was caused by a collision
(0.1-0.2) Sound depends on player actions and environment: moving through sand generates a different sound than moving through puddles)

| | | | |
|---|---|---|---|
| **Stage 7 - saving the game (recommended: 0.5, maximum: 1.5)** | | | - |

**Save and reload game (recommended: 0.5, max: 1.5)**
(0.1) Create a class derived from SaveGame
(0.2) Option to save the game during gameplay to a file without going through the main menu. For example, to a combination of keys or to click on special elements in the game.
(0.2) Extra points are given if when saving the game the user is asked by a widget if they want to save over the current file corresponding to the current game session or want a new file in which case they can opt for part of the name (eg save is of the form [username][timestamp][username]
(0-0.4) Saving relevant information for restarting the game from the remaining point and displaying the settings already made by the user. (Each different type of data from those listed 0.05). Scoring is only given if the memorized information is relevant to saving the game:

- Integer (integer or Integer64)
- Boolean
- Rational (Float)
- Character string (String or Text)
- Vector (eg for storing a color, location, etc.)
- data table (Array) of any type
- set (Set) of any type
- map (Map) of any type

(0.2) Create a function in Blueprint to read the save file and set some data in the game.
(0.2) AutoSave option - will save the game every time interval t.
(0.2) Saving at checkpoints. If the user achieves something special such as reaching a location or finding an artifact or defeating an enemy, the game will automatically save and upon re-entering the game will start from the previous checkpoint.

| | | | |
|---|---|---|---|
| **Stage 8 - C++. Design patterns. Object Oriented Programming.** | | | |

| | | | |
|---|---|---|---|
| | | | |
| **Stage 9. Visual effects: particle systems, post-processing effects.** | | | |
| | | | |
| **Stage 10 - documentation (recommended: 0.5, maximum: 1.1p)** | | | - |
| **Documentation (recommended: 0.5, maximum: 1p)**<br>Note: write short and to the point, the number of pages does not matter, but the required information should appear below. A documentation of one page can be worth maximum and one of 10 pages can be worth 0.1-0.2... You can also write "telegraphically" while going through all the required sub-points.<br><br>It will include the following parts/chapters:<br>1. (0.05) First page with name, surname, group, exam date, and game title. at the bottom of the page, the name of the subject. A table of contents to the chapters (in case there are several pages). The names of the blue chapters are those written in bold, black color. The pages (if there are more) will be numbered.<br>2. (0.1) Detailed description of the game (not the original plan but just what you managed to implement. What the game does, what the story is (this part may also contain fragments of stage 0). May contain printscreens from the game for clarification<br>3. (0.05) Technical specifications: operating systems the game was developed for, how much disk space it takes up, approximate memory it needs, additional packages that should be installed, whether or not it requires a network connection, what kind of data it saves and approximately how much large can reach a save file, the version of Unreal Engine on which the game was developed.<br>4. (0.05) Thematic specifications (required for the eventual publication of the game): game category/categories, motivation for choosing the game theme, a list of descriptive tags, is the game single or multiplayer, game time (estimate how many hours the game can be completed; for unlimited games it will be specified that the playing time can be however long), the languages in which the game is available.<br>5. (0.1) Actions available. Control. What actions we have available in the game and with which peripheral input devices we can achieve them (mouse, keyboard, joystick, etc.), with what combination of keys and buttons. What shortcuts are there? What options does the user have to define their own mouse/keyboard event combinations for various actions<br>6. (0.1) Own classes. Description of the classes created within the game: their role, important properties, what methods they implement, how they are integrated into the game. What design patterns were implemented (and brief description of each design pattern).<br>7. (0.1-0.2) Algorithms. Description of the algorithms used (for example, algorithms for generating a structure, special algorithms for calculating scores, artificial intelligence algorithms that control pawns/characters in the game. Listing the elements implemented in C++. screenshots with blueprints or sequences from C++ code.Analysis of algorithm complexity and efficiency in terms of both time and memory.Scoring depends on number of algorithms, complexity and analysis mode.<br>8. (0.05-0.1) AI tools in Unreal. Description of artificial intelligence tools (within Unreal) used and how to apply them in the game.<br>9. (0.1) List of completed tasks.**Mandatory for presentation**.Here will be the requirements copied and pasted from the scale, which the student has completed. The list of requirements will be organized by categories and subcategories exactly like this scale. The score from the scale will be written next to each task, and the student will make an estimated sum of the points under each category but also at the end, for verification (the scale is large and complex and it is easy to look at something in the presentation, with this measure we can check if we didn't skip anything).<br>10. (0.05-0.1) List of utilities used. You will list all utilities used to create assets (models, textures, sounds, etc.) for the game. You will list the assets and briefly explain how you achieved them.<br>11. (0.05) List of used external packages. You will list all packs added to the game, all assets taken from other sources (images, and in what context you used them. You will also provide a link to each pack.<br>12. (0.05-1) Bibliography. You will list books, tutorials (written or video), forums where you picked up ideas, pieces of code/blueprint. If the source is online, you will also write the link. For each source you will specify what exactly you took from there. | | | |
| **Other scoring categories** | | | |

| | | | |
|---|---|---|---|
| **Algorithmic (recommended: 0.5, maximum: 2)** | | | - |
| **Algorithmic (0.1-2p)**<br>Up to a maximum of 2p is given on algorithms depending on their number and complexity.<br>An algorithm is scored according to:<br>● relevance in the game<br>● efficiency in time and space<br>● special data structures used (vectors, matrices, graphs, neural networks)<br>● the mathematical formulas involved<br>● interaction with the in-game environment, with the state of the player<br>● the number of tasks it performs<br>● scalability<br>● Note: generally those who like puzzle games or strategy games will have the maximum here. The course game for example would get about 1-1.5 because of the maze and the enemies. A simple shooter, with only health calculation, small upgrades, etc., takes about 0.5-0.7. A game where nothing really happens (the man also tried something so as not to say that he never opened Unreal in his life) takes 0.1 | | | |
| **Appearance (recommended: 0.5, maximum: 1)** | | | - |
| **Appearance, relevance and accessibility**<br>It will be scored taking into account the following<br>● Placing objects in the scene in a logical and orderly way (for example several trees grouped on a terrain area to form a forest and not suspended in the air or on the head of any character - unless the game's story requires them to be there. You will make all the trees flying in the game now right?<br>● The land has a natural, logical, neat, pleasant shape. For example, it's not just a flat field with three dimples placed where it just happened to be).<br>● Sky Sphere has a relevant texture (we don't have clouds and sun for a game set in space)<br>● Using appropriate colors (that don't bother the eye) and nice textures. Texts can be taken from the Internet as long as copyright is not violated and the source is credited.<br>● Materials match the mesh (eg an object we know to be metallic will have a metallic sheen).<br>● Models are suitable in shape and size.<br>● The scene architecture and chromatics match the theme of the game and the intended user category.<br>● The lighting of the scene is appropriate (for example if we are in a room where we are looking for something, have a light source to distinguish the objects (unless the story calls for it)<br>● The displayed widgets are easy to use. Texts do not overlap. The contrast between the color of the text and the background is appropriate (easily readable).<br>● The animations are decent, with smooth transitions, not too much to make the scene tiring. | | | |
| **Code organization (recommended: 0.5, maximum: 1)** | | | - |
| | | | |
| **Originality / creativity (recommended: 0, maximum: 0.5)** | | | - |
| **It can be given for:**<br>● **very interesting game ideas**<br>● **special implementations**<br>● **special, interesting resources (created by the student).** | | | |

| **Bonuses/Optionals** | | | |
|---|---|---|---|
| **Modeling (eg Blender, Unreal in modeling mode) (recommended:0, max:0.9)** | | | - |
| (0.1-0.3 per model; maximum 0.9 for 3 models) The models modeled by the student in Blender (or another modeling utility) are scored according to their complexity. Up to a maximum of 3 different models are scored (So present the most complex models; if the differences between the models are small, it is considered low degree of complexity). The student must show the proof of the creation of the models (for example the files created by the utility, print screens with the progress of the work, the evolution of the model along the way), the explanation of the way of realization in the documentation (what tools he used). | | | |
| **Sounds (recommended:0, maximum:1)** | | | |
| (0.1-0.2 per file; maximum 1p for 5 files) The sounds created/recorded by the student themselves are scored according to their complexity. in whatever utility he wants (eg Audacity). Up to a maximum of 5 different sounds are scored (So present the most complex files; if the differences between the sounds are small, it is considered low complexity). The student must show the proof of the creation of the sound files (for example the files created by the utility, print screens with the progress of the work, the evolution of the track along the way), the explanation of the way of realization in the documentation (what tools he used). | | | |
| **External packages (recommended:0, maximum:1.4)** | | | |
| **External packages (recommended: 0; maximum: 1.4)**<br>(0.1 per pack; max 0.3 for 3 packs) Adding an external pack and using at least one mesh/texture/material from it. Up to a maximum of 3 different packages are scored<br>(0.05-0.1) Changing the parameters (except for positioning for actors) of at least one element inserted from the external package. It is scored according to complexity.<br>(0.1-0.2 per class; max 1p for 5 classes) Extending a class of any kind within the package and changing parameters, adding new parameters/methods. It is scored according to complexity up to a maximum of 5 different modified classes | | | |