Game project part 7: make it awesome

Overview:

The final stage of your game project is to make your Game Project awesome! This is your chance to flex your creative muscles, show off your technical skills, and produce a polished game.

Complete the base game:

Firstly you need to make sure that your base game is fully functional and bug free. You will need to have implemented all of the following from the previous game projects:

- Player interaction
- Canyons and coins interaction
- A scrollable game world
- Score and lives counters
- Game over states

You’ll also be marked on the gameplay and aesthetics of your game. You can score highly on this category by:

- Creating a playable but challenging level design
- Creating coherent and aesthetically pleasing graphics and animations
- Making sure your game is bug free and that the player controls are well tweaked (e.g. the jump is well timed)

The code philosophy videos 'The debugger's mindset' in 4.3 and 'Testing' 6.5 will help you in your attempts to iron out any bugs.

Your code also needs to be competently implemented and well-presented. Review your code and check:

- Indentation is correct
- Whitespace (empty lines and spaces) is properly used
- Inline comments but NOT commented out code
- Consistent and logical variable naming
- Good code organisation

You should also check for:

- Implicitly declared variables, redundant variable declarations or assignments
- Appropriate use of local and global variables
- Array traversal is correctly implemented
- Data structures and function design follow the prescribed scheme or a more advanced scheme that is rational in its design.
- Conditional logic is clearly and plainly implemented.
- Game elements are correctly anchored

If you’re not sure how to do this, then a good idea is to rewatch the code philosophy video 'The elegant coder' in 3.4 and also '12 top tips for naming variables' in 3.2.

We will reward students who want to include advanced techniques such as ES6 syntax and modularisation of code into multiple files. NB. using advanced techniques is an optional extra for those students who feel comfortable doing so. Whichever techniques are applied, they need to be used consistently in a way which demonstrates understanding.

Implement Extensions:

The next stage is to extend your code by adding up to three of the extensions which are demonstrated in the final topic:

1. Add sounds Use p5.sound to add sound effects to your game. Watch the tutorial video from the topic to do this. 2. Create platforms Use the factory pattern to create platforms. Watch the tutorial video from the topic to do this. 3. Create enemies Use a constructor function to create enemies. Watch the tutorial video from the topic to do this.

It’s up to you how many of the three you implement and to what degree you enhance the implementation from the basic tutorial code. The extension is graded with criteria for functionality, creativity and ambition. If you are unsure, then the grading rubric will guide you on how to scope your extension work.

Deliverables:

Hand in your final game project as a compressed folder in .zip format. It should contain the following.

- Your final game project code
- All assets needed to run the code (html files, library files, images, sound files etc.)
- A commentary of around 250 words in PDF format. This should describe:
    - your extension(s)
    - the bits you found difficult
    - the skills you learnt/practiced in completing the game project

Reading: End of term assessment – frequently asked questions

You are now working on your end-of-term assignment. Make sure you read the instructions carefully.

Before submitting, ensure you have completed the following:

1. Watched all the game project lectures up to now.
2. Attempted all the game project exercises up to now.
3. Completed all the ‘hack it’ activities and Sleuth activities up to case 802.
4. Review the feedback received for the mid-term project submission.

Additionally, make sure you either attend live or watch the recording of the end-of-term webinar run by the module instructor.

Here is also a list of frequently asked questions that can help you with the end-of-term submission:

Do we need to do extra things beyond the given instructions? What does it mean to ‘make an effort to go beyond the basic requirements’?

By exceptional criteria, we refer to programmatic aspects. Examples include, but are not limited to, conciseness in logic, technically interesting implementation of an idea, a certain finished quality to the code and the submission, and so on.

Does the project start have a time limit like an exam? The project has a deadline until which you can submit multiple tries. Please do not submit at the very last minute. Allow time to review your submission.

Is there a specific style of indentation that you would like us to follow?

Not really but the idea is for you to keep it consistent.

Can I add music to the game?

Yes.

Can I use external libraries?

It depends. We still recommend you build the programs as much from your own code as possible.

You will find all the instructions for submitting the end-of-term project in the following activity. Please take the time to read the requirements carefully, as not doing so can result in a loss of marks. Additionally, ensure that you do not submit at the very last minute; allow time for review.

Best of luck and create fantastic games!

Instructions

Overview

The final stage of your game project is to make your Game Project awesome! This is your chance to flex your creative muscles, show off your technical skills, and produce a polished game.

Complete the base game

Firstly you need to make sure that your base game is fully functional and bug free. You will need to have implemented all of the following from the previous game projects:

- Player interaction
- Canyons & Coins interaction
- A scrollable game world
- Score and lives counters
- Game over states

You'll also be marked on the gameplay and aesthetics of your game. You can score highly on this category by:

- Creating a playable but challenging level design
- Creating coherent and aesthetically pleasing graphics and animations
- Making sure your game is bug free and that the player controls are well tweaked (eg. the jump is well timed)

The code philosophy videos "The debugger's mindset" and "Testing" will help you in your attempts to iron out any bugs.

Your code also needs to be competently implemented and well-presented. Review your code and check:

- Indentation is correct
- Whitespace (empty lines and spaces) is properly used
- inline comments but NOT commented out code
- Consistent and logical variable naming

- Good code organisation

You should also check for:

- Implicitly declared variables, redundant variable declarations or assignments
- Appropriate use of local and global variables
- Array traversal is correctly implemented
- Data structures and function design follow the prescribed scheme or a more advanced scheme that is rational in its design.
- Conditional logic is clearly and plainly implemented.
- Game elements are correctly anchored

If you’re not sure how to do this, then a good idea is to rewatch the code philosophy video “The elegant coder” and also “12 top tips for naming variables.”

There are also some bonus points for those students who are able to include advanced techniques such as ES6 syntax and modularisation of code into multiple files. NB. using advanced techniques is an optional extra for those students who feel comfortable doing so. Whichever techniques are applied, they need to be used consistently in a way which demonstrates understanding.

Implement extensions

The next stage is to extend your code by adding up to three of the extensions which are demonstrated in the final topic:

1. Add sounds Use p5.sound to add sound effects to your game. Watch the tutorial video from the topic to do this.

2. Create platforms Use the factory pattern to create platforms. Watch the tutorial video from the topic to do this.

3. Create enemies Use a constructor function to create enemies. Watch the tutorial video from the topic to do this.

It’s up to you how many of the three you implement and to what degree you enhance the implementation from the basic tutorial code. The extension is graded with criteria for functionality, creativity and ambition. If you are unsure, then the grading rubric will guide you on how to scope your extension work.

Deliverables

In the first upload area, please submit your final game project as a compressed folder in .zip format. It should contain the following.

- Your final game project code
- All assets needed to run the code (html files, library files, images, sound files etc.)

In the second upload area, please submit your commentary of around **250 words** in **PDF** format. This should describe:

- your extension(s)
- the bits you found difficult
- the skills you learnt/practiced in completing the game project

Review Criteria

Base application (25 marks)

- Base functionality [6]
- Code presentation [6]
- Code competency [6]
- Advanced code techniques bonus [3]
- Game play, level design and aesthetics [4]

Extension (12 marks)

functionality (4):

creativity (4):

Extensions - ambition (4):

Commentary and deliverables (3 marks)