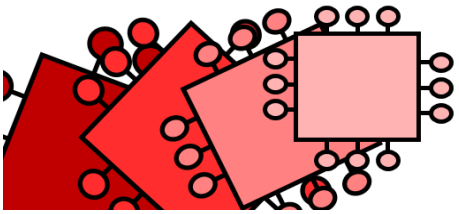


OCR Advanced GCE in Computer Science

H446-03/04

Unit 3

Project Advice

H446-03/04 – PROJECT ADVICE

CONTENTS

FORWARD BY THE CHIEF EXAMINER

It has been impressive in the past to see so many original ideas and obvious enthusiasm for developing coded solutions. It was also clear that the key to success lies in careful, thorough and detailed investigation of the problem and potential solutions.

Where candidates have gone beyond a simple interview and looked at the problem in greater depth the development became more focused and effective.

There have been a number of impressive coded games and activities including retro games like asteroids and teaching aids such as a binary tutor.

It is clear that when a student finds the topic area interesting they focus more on the task in hand and produce good quality solutions.

The best reports provided a commentary on the whole process in chronological order using segments of code, test results and further investigation to illustrate their progress towards a working solution.

It is worth noting that a number of centres have submitted electronic evidence alongside the printed report to good effect.

OCR will accept the work for this unit in electronic form and no longer require a printed report.

We are also quite happy to accept electronic forms of evidence for testing to supplement these reports, for example, avi files showing testing in progress.

GENERAL COMMENTS

OVERVIEW

The programming project is a major part of the A'Level course. It is worth 20% of your final grade.

You are required to demonstrate your ability to analyse, design, develop, test, evaluate and document a complete program written in a suitable programming language (see below for a list of languages you are allowed to choose from).

It is important that the nature of the problem you choose to solve allows you to demonstrate the full range of skills and techniques required in the mark scheme. Trivial problems, regardless of how well you solve them and write them up will not be able to provide you the right evidence (there is guidance below on the appropriate project ideas).

ADOPTING THE AGILE APPROACH TO SOFTWARE DEVELOPMENT

It is stated by the exam board that you are expected to *"Apply the computational approaches identified in Unit 2 to a practical coding problem and apply the principles of an agile development approach to the project development."*

This means that although you will probably do most of the initial requirements gathering, analyses and design work up-front it is **NOT** recommended that your project write up then follows discrete sections labeled up in the order of the assessment criteria. Development of a solution is an iterative process. You will tackle each part of your problem in turn, coding a procedure, module or function, testing it, modifying it then moving onto the next part. During the process of development you will regularly get feedback from your client, they will provide comments on how your solution is developing.

During development, due to problems, issues highlighted from ongoing testing or simply due to feedback from your user you might discover omissions or problems with your original requirements or design work. **DO NOT** go back and alter your requirements or designs to match, this is perfectly natural during development and the examiner expects to see this. Simply record any changes, new requirements, new algorithms, new tests in your development section as they appear.

This extended development section thus becomes a narrative on the process of producing your solution. This is known as "Telling the development story", it is the most natural way to capture evidence and is exactly what the examiners are expecting to see.

This will mean that evidence to support assessing your project might be found throughout your project report.

CHOICE OF PROGRAMMING LANGUAGE

The choice of which programming language to choose for your project is not a simple one. It will make sense to choose a language you are comfortable with, so we would suggest using the one you have been predominantly learning to program in or one similar to it. Your teacher / tutor will also be able to guide you in a choice of language for your project.

Whatever language you choose the exam board state that *"All tasks completed in all languages need to have a suitable graphical interface".*

They go on to specify the programming languages OCR will accept. These are:

- Python
- C family of languages (C# C+ etc.)
- Java
- Visual Basic
- PHP
- Delphi
- Monkey-X (Also now approved by OCR and excellent for games development)

This list should allow you to develop most programs you have in mind. For example, if you wanted to create a mobile phone app for an Android phone this could be done in Java. If you wanted to create it for an iPhone this could be done in Object C.

If you would like to use a programming language not on the list above, it is essential that your teacher / tutor contacts OCR and uses the consultancy service to get approval.

Note: Simple programming environments often used to introduce programming at KS3 such as Kodu, Scratch, Gamemaker are not appropriate for A Level.

PROJECT IDEAS

Your choice of program for your project is one of the most important choices you will have to make. Essentially you can solve any problem you wish, however it is vitally important that you get the scope and level of complexity right.

When deciding on your choice of problem think carefully about what data it might need. Most programs tend to use data in some way, such as a stock control system, booking system, online revision tool, a testing program, a retro computer game. These types of problems allow you to easily show evidence of creating, writing, reading, updating and deleting data from a file.

The problem you choose to solve must be sufficiently complex that you will be able to meet all the requirements of the mark scheme, but not overly ambitious so as to become unachievable. Remember this is only A'level. You will not be writing CoD: Modern Warfare! It is always better to have a small program well done and well written up than a huge problem you can't solve.

Ultimately your teacher / tutor will guide and have to approve your choice of problem to solve. However here are some suggestions of the sorts of problems which would be appropriate along with the types of languages you could use to solve them.

DESKTOP DATA HANDLING PROGRAMS

Programs which allow you to manage a database of information via a graphical user interface. Such as a student's records system, a car dealership database or theatre booking program. These can often have limited coding elements, however there is lots of scope to add validation checks and discuss data structures.

Suggested languages: VB, Python, C#, C++, C, Delphi

WEB BASED DATA HANDLING PROGRAMS

These are often similar problems to the desktop ones except that interface is constructed to run in a Web-Brower. Although this is a much more common and modern approach in our online connected world it does come with its problems. Web standards are constantly changing. You will often need to know more than more language e.g. Javascript for the client-side and PHP/ASP for the server side, and possible some HTML / CSS for the web based interface! You will also need to think about how your system will be hosted. If you are not using a paid online hosting provider you will need to get your schools IT technicians to host it for you.

Suggested languages: HTML5, JavaScript, SQL, Java, jquery, VB/ASP.net

DESKTOP GAMES

These types of problems typically allow for plenty of computationally complexity but can be really fun to do. The major problem here is biting off more than you can chew! Keep it simple. If you get completed before the deadline you can easily add extra functionality like new levels, high score tables, power-ups etc.

Suggested languages: ActionScript, C#, C++, XNA, pygame, Java, Monkey-X (**fully supported at craigndave.org**)

MOBILE GAMES / APPS

Any game you want to design for a desktop system can easily be adapted and made for a modern touch screen mobile phone in the form of an APP. On the plus side the sort of IDE's that assist with mobile app development usually require less computational complexity but you often end up having to learn how to use a new IDE from the one you have been used to using.

Suggested languages: Objective-C (IOS), Java, Python (via SL4A) for Andriod, Java for Blackberry, HTML for Windows Phones, Django (based on Python), Monkey-X (**fully supported at craigndave.org**)

WEB BASED GAMES

Similar to other games, except they are designed to run entirely through a web-browser.

Suggested languages: HTML5, ActionScript Visual Studio (using Silverlight), Monkey-X (**fully supported at craigndave.org**)

THE USER

Every project needs an identified user or target audience. They play a key role in an OCR Computing project.

As far as possible you should try to get a real user who will be able to have regular contact with. For example if you are writing a Chemistry revision program for 15 year olds then you could choose both a Chemistry teacher at your school and a student 15 years of age. They would be able to give you feedback and requirements on your solutions from the two required user perspectives and it would be easy to get hold of them during school.

For the purposes of some project development, your teacher/tutor can be a pseudo-user: someone acting on behalf of a real user. For example, if you are making a game for young children or a retro computer game, it may be difficult to have regular contact with a user. Your teacher can be that person in these cases.

ABOUT THIS DOCUMENT

What follows is the mark scheme for each section of the report write-up taken word for word from the specification (in black text), together with comments about that section from the external examiner (in blue text). Comments for markers are next which may help in understanding how marks are awarded within the section (in green text). Advice from teachers is the final part in each section (in red text).

Not all projects will necessarily easily fit the mark scheme so these comments inform you how to tackle each section.

Each section also comes with an evidence Do's & Don'ts checklist.

ANALYSIS (MAX 10 MARKS)

1-2 marks

- Identified some features that make the problem solvable by computational methods.
- Identified suitable stakeholders for the project and described them and some of their requirements.
- Identified some appropriate features to incorporate into their solution.
- Identified some features of the proposed computational solution.
- Identified some limitations of the proposed solution.
- Identified some requirements for the solution.
- Identified some success criteria for the proposed solution.

3-5 marks

- Described the features that make the problem solvable by computational methods.
- Identified suitable stakeholders for the project and described how they will make use of the proposed solution.
- Researched the problem looking at existing solutions to similar problems identifying some appropriate features to incorporate into their solution.
- Identified the essential features of the proposed computational solution.
- Identified and described some limitations of the proposed solution.
- Identified most requirements for the solution.
- Identified some measurable success criteria for the proposed solution.

6-8 marks

- Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach.
- Identified suitable stakeholders for the project and described them and how they will make use of the proposed solution and why it is appropriate to their needs.
- Researched the problem in depth looking at existing solutions to similar problems identifying and describing suitable approaches based on this research.
- Identified and described the essential features of the proposed computational solution.
- Identified and explained any limitations of the proposed solution.
- Specified the requirements for the solution including (as appropriate) any hardware and software requirements.
- Identified measurable success criteria for the proposed solution.

9-10 marks

- Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach.
- Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs.
- Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.
- Identified the essential features of the proposed computational solution explaining these choices.
- Identified and explained with justification any limitations of the proposed solution.
- Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements.
- Identified and justified measurable success criteria for the proposed solution.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

In this analysis phase you need to make sure to include all of the following sections:

1. Defining the problem & the Stakeholders

Start by giving a brief background to the problem. Answer the questions:

What is the company?
What does the company do?
Who are the stakeholders / end users?
What problem do they have?
How will they make sure of your proposed solution and why is it appropriate to their needs?

This initial description of the problem should be no more than a couple of paragraphs.

If you are writing a computer game, give a description of the type of game it is with a very brief explanation. Explain who will play the game and on what platform. Identify the key requirements outside the actual game play itself, e.g. in the case of a mobile phone game: easy to pick up and put down, pause at any point and continue later.

If the problem has already been solved, pretend you are solving it for a new platform or user. E.g. with space invaders, perhaps it is a mobile phone version or a new twist on the old concept.

Note about stakeholders: Make sure you clearly name all of the stakeholders / users for your system. These must be actual named individuals that you can have regular contact with as they will be required to give you feedback and interviews throughout the development of your project. You can have more than one stakeholder / user. For example if you are creating a Maths revision utility for 11 year olds then you would clearly have two users, a Maths teacher and an 11 year old students. They will both be able to give you requirements and feedback from their different perspective.

It is also acceptable to have chosen a "persona", someone who personifies the typical user for your chosen system. This will be most likely if you choose to make a game. Decide who your game is targeted at e.g. "Teenagers into mobile gaming" and then choose a named person from this target group who you will be able to have regular contact with to act as your stakeholder / end user.

Make sure in this section to not just simply list our users / stakeholders. For top marks you must make sure to explain how they will make sure of your proposed solution and explain why it is suitable for their needs.

2. Justification of how the problem can be solved by computational methods

You must fully justify how the solution you wish to program can be solved by computational methods. These are all the methods you have been studying for Unit 2 and include:

- Thinking Abstractly & Visualisation
  - How will your problem simplify reality? If you are producing a game, simulation, training aid, booking system etc what detail **IS** important and what details from reality will you ignore or omit?
- Thinking Ahead
  - What data / inputs will be required for your solution to work?
- Thinking Procedurally & Decomposition
  - Can your problem be easily broken down and tackled in smaller chunks?
- Thinking Logically
  - Will your problem have obvious decisions points for branching or repetition (looping)?
- Thinking Concurrently
  - Will there be any parts of your problem which could be solved or could happen at the same time?

3. Research

In this section you are describing the problem. With a game, take this approach to the write up:

1. Initial research: start by identifying a similar game (perhaps from the internet) and describe the mechanics of how it plays.
2. Form a set of questions to ask the user about how your game should look, sound and play. Document the user responses to these questions. (See note 1) E.g.
   Q: What does the player control in the game?
   A: The player controls a spaceship that can move left and right at the bottom of the screen.
3. Deliberate on the answers you are given and the initial research. This will inform the proposals.
4. Propose a solution to the problem by describing each element of the game in detail. You can have mock ups of the graphics from a drawing application at this stage.
   (See notes 2 & 3)
5. Get a response from the user about whether this meets their expectation.
6. Get an agreement from the user.

Note 1: You need to conduct an interview and/or observation of at least one existing system to know the details of what you need to know to make the program later. Keep records of the questions and observations you make, together with answers to questions.

Note 2: You need to discuss in detail exactly what the system is going to do, but not how it is going to do it. This is not about design or algorithms, it's about the requirements. Here we are focusing on the what, not the how. Detail is very important in this section in your descriptions of the system.

Note 3: Consider a typical space invaders game. You would need to discuss that the player controls a ship. The ship can move left or right inside a fixed plane at the bottom of the screen. The ship can fire one bullet at a time. There can't be more than one bullet from the player on the screen at the same time. The objective is for the player to shoot all the invaders. The invaders start towards the top of the screen and move from left to right together in initially 5 rows of 12 invaders. When the right-most invader reaches the right edge of the screen, all the invaders move down a little on the screen and start moving from right to left. When the left-most invader reaches... etc.

'Leave no stone unturned'. Your analysis should include sufficient detail so if you were to get a programmer to read the analysis, there is nothing more they would need to ask before making the solution. Of course the really fine details may not be entirely known and will be picked up in the development process. For example, the speed

at which the ship moves across the screen. You would need to play the game to know what feels right. That is unlikely to be known at the analysis stage and the necessary dialogue between you and a user will gain you marks in the design section later.

You should write your analysis as if you were having a discussion with a user. For example, "The intended audience for the game is..." "I am using my teacher as a representative for that audience." "I discussed the requirements of the game with..." "It was suggested to me that..."

Whatever the problem, it will always have a target audience and therefore an identifiable user which you should be discussing requirements with and keeping records of these discussions.

4. Features of your proposed solution

In this part you should make sure to clearly explain each of the features of your proposed solution. How you choose to do this is up to you, however look carefully through your research and analysis and make sure you have not missed anything.

In this section you should also identify any limitations of your proposed solution. It will, by the nature of an A'Level project be limited. If it is a game what won't it do, be realistic. This is a good time to flag up desirable features that will not be included in the solution (you can revisit this again when you write your evaluation at the end).

5. Hardware & Software requirements

You should discuss the hardware and software required to run your program. E.g. an IBM compatible PC with x processor , y memory and z hard disk space running the Visual Basic runtime libraries? Find out the necessary spec to run the development environment i.e. VB or Access on a computer.

If any additional software is required to run your solution or if your solution is only intended to work with specific versions of software this needs to be identified here.

6. Success criteria / Requirements specification

As a summary of the analysis, create a numbered point list or table of the exact and actual success criteria / requirements. Call this the "Success Criteria / Requirements Specification." Avoid requirements that cannot be measured. E.g. "It must be easy to use" is too vague. "The user should be able to find a product within 20 seconds" is better. "A player scores 50 points for each invader." Remember, specific and measurable. It should contain numbers. Numbers of records, users, invaders, points etc.

EVIDENCE DO'S AND DON'TS CHECKLIST

This section of your project write up **must** include:

SECTION	DO's		DON'Ts
A description of the problem	1 Provide an outline of what your problem is 2 Provide an explanation of features required in your computer program to provide a solution to solution to your problem	☐ ☐	Reply on simple statements of the problem
Identify all stakeholders	3 Identify all the stakeholders (users) as individuals, groups or persona 4 Keep returning to the stakeholders (users) for input throughout the project	☐ ☐	Identify an end user who cannot be easily contacted throughout the project

Justify why the problem can be solved by computational methods	5 Explain why the problem is suited to a computer program 6 Explain the features of your problem that are amenable to a programmed solution 7 Explain why the output from the solution is valuable to the stakeholders (users)	☐ ☐ ☐	Simply state that you are going to create a program because it is needed. You must justify decisions
Research	8 Provide detailed research into existing solutions to similar problems 9 Show that the research identifies features that can be adapted for use in your proposed solution 10 Show how the research provides insight into the proposed solution and how the features to be used are appropriate	☐ ☐ ☐	Rely on your own input for the solution to your problem Rely on an interview with and end user for all your research into the problem
Features of the proposed solution	11 Identify the features of your proposed solution 12 Identify any limitations of your proposed solution 13 Be realistic about what can be achieved in the time allowed	☐ ☐ ☐	Attempt to solve to solve problems that are too complex in the time allowed
Software and hardware requirements	14 Specify any hardware requirements for your solution 15 Specify any software requirements for your solution 16 Do identify any additional utilities that will be required to implement your solution	☐ ☐ ☐	List all the software available simply to justify a choice Simply identify what software you are using
Success criteria	17 Specify the success criteria (requirements) for your proposed solution 18 Do specify success criteria (requirements) that can be demonstrated through testing	☐ ☐	Specify vague subjective criteria, such as colorful interface or easy or quick to use

DESIGN (MAX 15 MARKS)

1-2 marks

- Described elements of the solution using algorithms.
- Described some usability features to be included in the solution.
- Identified the key variables / data structures / classes (as appropriate to the proposed solution).
- Identified some test data to be used during the iterative or post development phase of the process.

5-8 marks

- Broken the problem down systematically into a series of smaller problems suitable for computational solutions describing the process.
- Defined the structure of the solution to be developed.
- Described the solution fully using appropriate and accurate algorithms.
- Described the usability features to be included in the solution.
- Identified the key variables / data structures / classes (as appropriate to the proposed solution) and any necessary validation.
- Identified the test data to be used during the iterative development of the solution.
- Identified any further data to be used in the post development phase.

9-12 marks

- Broken the problem down systematically into a series of smaller problems suitable for computational solutions explaining the process.
- Defined in detail the structure of the solution to be developed.
- Described the solution fully using appropriate and accurate algorithms explaining how these algorithms form a complete solution to the problem.
- Described, explaining choices made, the usability features to be included in the solution.
- Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) explaining any necessary validation.
- Identified and justified the test data to be used during the iterative development of the solution.
- Identified and justified any further data to be used in the post development phase.

13-15 marks

- Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process.
- Defined in detail the structure of the solution to be developed.
- Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem.
- Described, justifying choices made, the usability features to be included in the solution.
- Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation.
- Identified and justified the test data to be used during the iterative development of the solution.
- Identified and justified any further data to be used in the post development phase.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

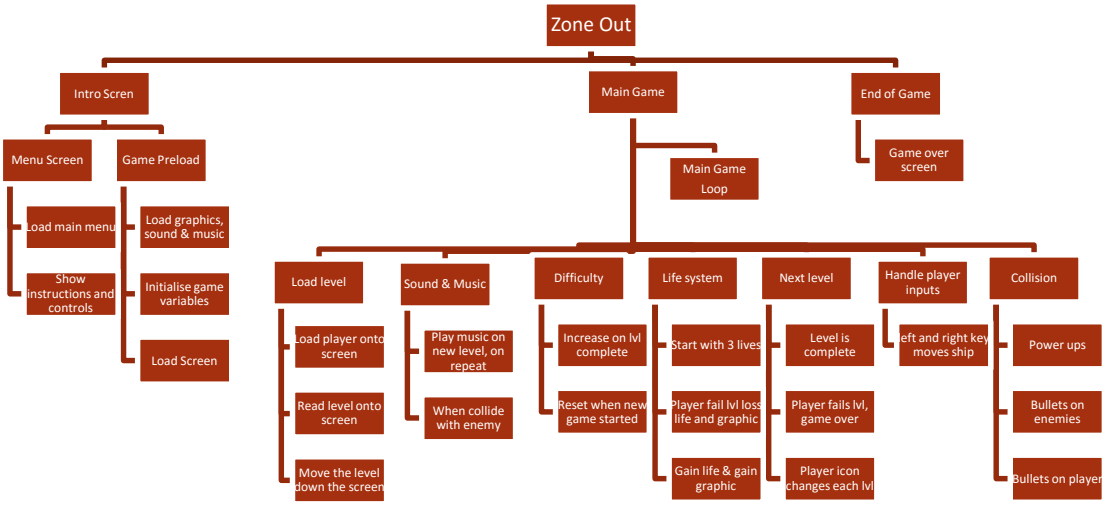
There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

In this section you are presenting your proposals for how you will solve the problem. You should include the following:

1. A systems diagram (Top down modular design)

It should identify the key stages of the solution. See the exemplar below:



2. Sub headings for each box from the systems diagram

   a. Proposed screen designs and usability features

Include sketches of what the screens or form designs will look like. These can be 'mock-ups' done in a Word document, Publisher, Excel etc. or hand drawn sketches scanned in. These should include the real graphics and dimensions of the graphics at this stage.

If there are output reports these should be planned too. Be careful to include all major screen designs. If you are making a game, each level should be sketched, together with the main menu, help screen, high score screen etc. . If your project is a database all the user interface forms and reports should be sketched.

   b. Detailed summary of the process including key variables and structures

Under the sub headings a detailed summary should be written of what happens at each stage. This would include some description of what coding/algorithms would be necessary. This is not algorithms themselves, but descriptions. Enough for an experienced programmer to derive pseudo code from. E.g. move the invaders 10 pixels according to the invader direction variable. +10 moves right, -10 moves left. If the invader has reached the

edge of the screen, reverse the values to cause the invader to move in the opposite direction next time the procedure is called.

Under the relevant sub-heading identify all the 'objects' required in the program if it is a game, such as the ship, invaders etc. You should include class diagrams for these.
It is **VERY** important in this section that you fully identify any key variables and data structures you intend to use.

Make sure to name the **procedures**, **functions**, **methods**, **classes**, **arrays**, **structures**, **records**, **files** etc. and any key **variables/constants** along with their **datatype**.
For example, a high score table may be stored in memory as an array and on a disk in a sequential file. For games with levels, include one example of a text file that will be read in for the level data.

In a database, the tables, fields, data types and relationships should be identified.
In a database solution under the relevant sub-heading identify all the queries, macros or code elements needed.

c. Test data for development

Identify appropriate data that you will be using during "The Development Story" to test the functionality of your program as it develops. This shows your ability to Think Ahead, the data you list here should be used as you develop your solution for the purpose of proving it is evolving as expected.

Note: This is **NOT** a test plan at this stage. There is no requirement to create a full test plan yet. This is simply the data you will be using at each stage of the development process.

3. Algorithms

This is where the real thought about the internal workings of your solution begins. This section is often over looked and performed poorly by candidates.

You may come back to this section later to explain anything complex about your solution. For example, your collision detection may use a pruning tree technique to speed up the checking process. That sort of detail ought to be explained here but is likely to be done once you have written the code.

You can produce your algorithms in any format you wish such as pseudo code or flow diagrams.
It is important you show how your various algorithms fit together to form a complete solution to the problem.
The best candidates will perform some dry-run testing here to prove up front in their design that they will work as required when coded.

4. Explain the traditional SDLC process

You should explain that you are taking a traditional approach to the stages of the Software Development Life Cycle for the Analysis and Design phases. However, once you start developing you will be taking an agile / iterative approach. Make sure you justify this approach to your development. You should make it clear that you will have serious amounts of user involvement and testing all the way through the development of your solution and not simply bolted onto the end of your project.

5. Test data for beta testing

Explain you are going to test the program as it is developed, but will have a black box approach to the final beta / acceptance testing.
Identify here, and justify, any test data you intend to use post-development to ensure that your completed solution meets the success criteria written up in your requirements specification.

The best candidates here will identify not just data that is designed to work but also data that is designed to break their program. Try to identify **valid**, **borderline** and **invalid** data where possible so that it can be seen that upfront you are thinking about the robustness of your finished solution, good test data should attempt to break the system!

Note: Once again, you should **NOT** be creating a full blown test plan at this stage. The data you specify here will however be used in a final test plan for the finished product during the post-development stage after the main bulk of "The Development Story".

6. Sign off the proposal

You should show your proposed screen designs to your user as well as talking them through your solution. Record some feedback from them at the end of your design section and show that they are agreed in principle, subject to any changes that need to be made before development commences. Get a dated signature at the bottom of this section to prove you did it.

EVIDENCE DO'S AND DON'TS CHECKLIST

This section of your project write up **must** include:

SECTION	DO's		DON'Ts
Decompose the problem	19 Provide evidence of decomposing your problem into smaller problems 20 Provide evidence of a systematic approach, explaining and justify each step in the process	☐ ☐	Simply state the problem as a single process
Structure of the solution	21 Provide a detailed overview of the structure of your solution	☐	
Algorithms	22 Provide a set of algorithms to describe each of the sub-problems 23 Show how the algorithms fit together to form a complete solution to your problem 24 Show how the algorithms have been tested to show that they worked as required	☐ ☐ ☐	Simply provide an outline data flow Provide code or reverse engineered code as an algorithm
Usability features	25 Describe with justification the usability features of your proposed solution 26 Explain and justify the design of any user interface / screen designs or interfaces with other systems	☐ ☐	Spend ages creating colorful diagrams of the user interface
Key variables and structures	27 Identify and justify the key variables 28 Explain and justify the data structures that are to be used in your solution 29 Describe and justify any validation required	☐ ☐ ☐	
Test data for development	30 Identify and justify any test data to be used during development (this is appropriate test data that can be show to test the functionality of your program for development testing purposes)	☐	Create a full test plan for this stage; this is data to be used at each stage of the development process
Test data for beta testing	31 Identify and justify test data to be used post-development to ensure the system meets the success criteria (requirements) 32 Identify data that is designed to test the robustness of the solution; good testing should attempt to break the program	☐ ☐	Create a test plan for this at this stage; the data will be used in a final test plan for the product at the post-development testing stage

DEVELOPING A CODED SOLUTION (MAX 25 MARKS)

ITERATIVE DEVELOPMENT OF A CODED SOLUTION (MAX 15 MARKS)

1-4 marks

- Provided evidence of some iterative development for a coded solution.
- Solution may be linear.
- Code may be inefficient.
- Code may not be annotated appropriately.
- Variable names may be inappropriate.
- There will be little or no evidence of validation.
- There will be little evidence of review during the development.

5-8 marks

- Provided evidence for most stages of the iterative development process for a coded solution describing what they did at each stage.
- Solution will have some structure.
- Code will be briefly annotated to explain key components.
- Some variable and/or structure names will be largely appropriate.
- There will be evidence of some basic validation.
- There will be evidence that the development was reviewed at some stage during the process.

9-12 marks

- Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did at each stage.
- Provided evidence of some prototype versions of their solution.
- The solution will be modular in nature.
- Code will be annotated to explain all key components.
- Most variables and structures will be appropriately named.
- There will be evidence of validation for most key elements of the solution.
- The development will show review at most key stages in the process.

13-15 marks

- Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why.
- Provided evidence of prototype versions of their solution for each stage of the process.
- The solution will be well structured and modular in nature.
- Code will be annotated to aid future maintenance of the system.
- All variables and structures will be appropriately named.
- There will be evidence of validation for all key elements of the solution.
- The development will show review at all key stages in the process.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

Now your analysis and design are finished it is time to start development. This will be done in an agile way, that is to say that you will tackle each part of your problem in turn, coding a procedure, module or function, testing it, modifying it then moving onto the next part. During the process of development you will regularly get feedback from your client, they will provide comments on how your solution is developing. This process is known as telling **"The development story"**.

During your development section of your report you must make sure to include the following details:

- Name of the stage e.g. player movement.
- Date of each part of development
- Commented, indented and annotated code snippets / segments
- Explanation of testing being carried out and modifications to make
- Screen shot of output
- Regular input and feedback from your stakeholders / users

You will be able to gain marks for many other sections of your report such as "Testing" and "Evaluation" under this section if you do it well. You **MUST NOT** simply leave any mention of testing & evaluation until after you have finished coding, this is an unrealistic way to develop a solution and the examiner will be aware of this!

Although you will have snippets of code throughout your project report to help you tell the development story your entire code listing should be printed and fully annotated and should be included as an Appendix called "Code Listings" at the back of your project report.

It is vital as you tell "The Development Story" to make sure you:

1 Provide prototyping versions of your program at each stage of the process, make sure to include annotated code snippets which explain what has been done.
2 Provide evidence of testing each part as it is developed using the test data you identified back in your design.
3 Provide evidence of any validation you are using.
4 Get your user in at regular points to review how your solution is developing.
5 Explain any problems / issues that arise as they arise in an honest way.
6 Explain any changes / modifications which arise to your original design. **DO NOT** go back and alter your analysis or design. It is fine to come up with new algorithms, requirements, screen designs as long as there is justification. Credit will be given in the mark scheme for this.

TESTING TO INFORM DEVELOPMENT (MAX 10 MARKS)

1-2 marks

- Provided some evidence of testing during the iterative development process.

3-5 marks

- Provided some evidence of testing during the iterative development process.
- Provided evidence of some failed tests and the remedial actions taken.

6-8 marks

- Provided evidence of testing at most stages of the iterative development process.
- Provided evidence of some failed tests and the remedial actions taken with some explanation of the actions taken.

9-10 marks

- Provided evidence of testing at each stage of the iterative development process.
- Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

This should **NOT** be a separate section in your report, this should form an integrate part of "The Development Story". You can therefore see your development story as providing you with evidence for all of the 10 marks above. Read the mark scheme carefully above for "Testing to inform development" and make sure this is covered during "The Development Story".

EVIDENCE DO'S AND DON'TS CHECKLIST

This section of your project write up **must** include:

SECTION	DO's		DON'Ts
Iterative development	33 Provide evidence of iterative development showing how the complete program was developed stage by stage "The development story"	☐	Simply supply completed code for the program as evidence
	34 Provide evidence showing how each section of the program was coded and tested	☐	
Prototyping	35 Provide prototype versions of the program at each stage of the process that show the annotated and explained code	☐	
	36 Provide evidence of testing at each stage using the test data identified in the design section	☐	

Annotated modular code	37 Annotate the code at each stage of the process 38 Use meaningful names for all variables, structures and modules 39 Provide code in a modular form 40 Provide the code as separate modules	☐ ☐ ☐ ☐	Simply supply the complete code for the program as evidence; the code must be developed in suitable stages
Validation	41 Supply evidence of validation 42 Supply evidence that the validation has been tested and works as expected 43 Supply evidence that all testing covers a wide range of valid and invalid inputs and situations	☐ ☐ ☐	
Reviews	44 Review each stage of the process in the development phase, summarizing what has been done and how it was tested 45 Explain any changes required and any modifications to the design of the solution that result from the testing	☐ ☐	

EVALUATION (MAX 20 MARKS)

TESTING TO INFORM EVALUATION (MAX 5 MARKS)

1 mark

- Provided evidence of some post development testing.

2 marks

- Provided evidence of final product testing for function.

3-4 marks

- Provided annotated evidence of post development testing for function.
- Provided annotated evidence for usability testing.

5 marks

- Provided annotated evidence of post development testing for function and robustness.
- Provided annotated evidence for usability testing.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

This is the stage now where you create your full test plan and carry out your final set of acceptance testing with your stakeholders / users.

Create a table showing everything that needs to be tested by your users to be assured the system or game works as intended.

The tests should include: **valid** inputs, **invalid** inputs and **extreme** cases. It also needs to check all the 'events'. E.g. "When a bullet touches an invader".

Test No.	What is being tested	Input data or actions	Expected outcome	Actual Outcome
1	The player's ship cannot leave the screen to the left or right.	Move ship to far left and hold left key down. Move ship to the far right and hold right key down	Ship stops and no longer responds to key press	As expected
2				

Record in the actual outcome what actually happens when the test is performed. Ask the user to sign to confirm the program works as intended.

See this section as the beta-testing phase, carried out by a test team or the user rather than the programmer. The program should be complete when you tackle this section.

Evidence that the program works to the test table is very important. This evidence can be before and after screen shots, but could also be a video of the game being played (covering all the tests), captured using software. If you use a video, submit it together with your write-up.

EVALUATION OF SOLUTION (MAX 15 MARKS)

1-4 marks

- Commented on the success or failure of the solution with some reference to test data.
- The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.

5-8 marks

- Cross referenced some of the test evidence with the success criteria and commented on the success or otherwise of the solution.
- Provided evidence of usability features.
- Identified some limitations on the solution.
- The information has some relevance and is presented with limited structure. The information is supported by limited evidence.

9-12 marks

- Used the test evidence to cross reference with the success criteria to evaluate the solution identifying whether the criteria have been met, partially met or unmet.
- Provided comments on how any partially or not met criteria could be addressed in further development.
- Provided evidence of the usability features.
- Considered maintenance issues and limitations of the solution.
- There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.

13-15 marks

- Used the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria has been fully, partially or not met in each case.
- Provided comments on how any partially or unmet criteria could be addressed in further development.
- Provided evidence of the usability features justifying their success, partial success or failure as effective usability features.
- Provided comments on how any issues with partially or unmet usability features could be addressed in further development.
- Considered maintenance issues and limitations of the solution.
- Described how the program could be developed to deal with limitations of potential improvements / changes.
- There is a well developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.

External Examiners Advice

There is currently no external examiners advice on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance from the exam board.

Notes to markers

There is currently no external advice to markers on the Unit 3 Computing Project as this is the first series of the A'Level qualification. After the 2017 examiners report this section will be updated to reflect all external advice and guidance given to markers from the exam board.

Teachers advice

Your evaluation can take many forms. It can be a written report, interviews, tables with cross-reference links. Video evidence or a combination of these. However you choose to structure your evaluation it must make sure to cover **ALL** of the following areas in detail:

1 Comments on how well the solution matched the original requirements
2 Comments on any changes that you had to make to the original design during "The Development Story"
3 Comments on any unmet criteria / requirements and how these might be tackled in the future
4 Comments on any additional features that might be useful to your solution and these might be approached in the future
5 A discussion on future maintenance of your program
6 A discussion on any limitations of the current version of your program
7 Comments on the maintenance features included in your program.

EVIDENCE DO'S AND DON'TS CHECKLIST

This section of your project write up **must** include:

SECTION	DO's		DON'Ts
Testing	46 Provide evidence of testing on the completed solution	☐	
	47 Provide evidence that the system functions as designed	☐	
	48 Provide evidence that the system is robust and will not fall over easily	☐	
	49 Cross-reference the test evidence against the success criteria (requirements) from the analysis section to evaluate how well the solution meets these criteria	☐	
Usability features	50 Show how the usability features have been tested to make sure they meet the stakeholder's (user's) needs	☐	
Evaluation	51 Comment on how well the solution matches the requirements	☐	Comment on the development process and anything your learned or how much you enjoyed it
	52 Comment on any changes that were made to the design during the development stage	☐	
	53 Comment on any unmet criteria or features that might be useful and how these might be approached	☐	
Maintenance	54 Discuss future maintenance of the program and any limitations in the current version	☐	
	55 Discuss how the program might be modified to meet any additional requirements or changing requirements	☐	
	56 Comment on the maintenance features included in the program and report	☐	

H446-03/04 (UNIT 3) PROJECT HAND IN CHECKLIST

1. Make sure you project as a title page with:
    a. "Title of Project"
    b. The wording H446-03
    c. Your full name
    d. Your candidate number
    e. Your centre name & number

2. Make sure your project contains a contents list

3. Make sure every section of your project is included and clearly labelled:
    a. Analysis
    b. Design
    c. Developing the coded solution ("The Development Story")
    d. Evaluation

4. Make sure every single page is clearly numbered

5. Make sure your project as a bibliography

6. Make sure one of your Appendix's is clearly labelled as "Code Listings"
    a. This should include a full print out of **ALL** code in colour

7. Make sure you have included a copy of your entire solution on a CD, DVD or Pen Drive

INITIAL PROJECT IDEAS

Use this area to record your initial ideas / thoughts for your Computer Science coursework project.

Your ideas will be used as the basis for writing the start of your “Analysis”.

NOTES

H4406-03/04 COMPUTER SCIENCE COURSEWORK TRACKING

Bring this along to each check point meeting, it will be filled out with your teacher.

Check Point Date	Notes on progress required for next checkpoint

A'Level OCR Computer Science H4406-03/04 Coursework Guide

For more help and resources, including appropriate project tutorials and guides for the Computing Project please visit craigndave.org