TINKERING AUDIO

Version 3.1
Computing
COMP120

Dr Michael Scott

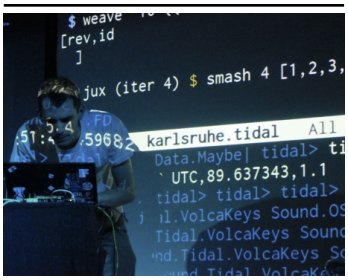
Introduction

"The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures"

— Fred Brookes

"Students come into programming classes with a broad range of backgrounds—some have experience in several programming languages, others have never programmed before in their life! Being able to engage with the community and support each other is important."

— Robert Macredie



Tidal is an algorave language. It allows you to code musical patterns live using text, describing sequences and ways of transforming and combining them, exploring complex interactions.

Creative computing encompasses the broad realms of digital media, computer programming, and human-computer interaction. It is important to draw these areas together in an applied way to develop the foundational skills required of all computing professionals. You will, therefore, leverage the principle programming constructs of representation, operation, sequence, branching, iteration, and functional decomposition to exercise your creativity. You will also start to leverage more advanced data structures and object-orientated programming principles. This will prepare you to tackle challenges in many creative domains.

In this assignment, you are required to complete a series of activities which culminate in a demonstration of a computer program that you have written in C#. This computer program will address a "contract" which will require you to *tinker* with a digital audio in a creative way according to given contextual cues, using the techniques that you have learned through the activities.

This assignment is formed of several parts:

(A) **Propose** your response to the contract **and outline** your goals:
- i. **setup** an appropriate repo to host the project;
- ii. **document** which contract you will work on and **list** the requirements it implies in a README;
- iii. **declare** the terms of your software license in a LICENSE;
- iv. and **associate** your repo with this assignment via LearningSpace.

(B) **Complete** the quizzes:
- i. **answer** the multiple-choice questions;
- ii. and **give** appropriate responses to the code writing exercises.

(C) **Report** your progress with your draft program:
- i. **outline** the **FIVE** algorithms you intend to implement;
- ii. **suggest** best practices to improve future maintenance;
- iii. **justify** the terms of your software license;
- iv. and **illustrate** appropriate versioning practices.

(D) **Review** the draft programs of your peers:
- i. **share** your draft program for review;
- ii. **scrutinise** the work of your peers and **suggest** improvements;
- iii. **review** any issues raised by your tutor and/or your peers;
- iv. and then **update** your draft program.

(E) **Submit and demonstrate** your final program:
- i. **show** your academic integrity;
- ii. **justify** decisions made during the project;
- iii. as well as **evidence** your individual programming knowledge.

Important Note: Unlike the previous assignment, you will work collaboratively; typically, in a pair or as a small mob. This is to help you to develop your collaborative programming skills. Please carefully review the assessment rubric at the end of this brief.

Assignment Setup

This assignment is a **programming task**. Create a repository using the template below. Extend the repository with sub-directories and modify the .gitignore as appropriate. Create the README.md and LICENSE.md files.

https://github.falmouth.ac.uk/Games-Academy/blank-unity-project

Please note that there should be **ONE** repository for the group, which you share. You may have to set permissions on your repository to allow other students to push code to your repo.

Please remember to create in repository in the correct organisational area on our GitHub Enterprise Server (i.e., Student-Work) and to use the appropriate naming convention which includes the module code and your student identification numbers (i.e., comp120-2-0001210-0103255).

Once the repo has been created, please connect your repository to your assignment by pasting the .git link in the assessment submission area on LearningSpace.

You **MUST NOT** add any other text to the submission field.

You each invidually should check that the repo is correctly associated with this assignment on LearningSpace.

Part A

Part A consists of a **single formative submission**. This work is **collaborative** and will be assessed on a **threshold** basis.

To complete Part A, follow the assignment setup instructions, and then write about your contract in the relevant LearningSpace activity. You are to select and interpret **ONE** contract together with your partner and propose your response to it.

Show this to your tutor in your timetabled tutor meeting. If acceptable, this will be signed-off. Then, start programming.

You will receive immediate **informal feedback** from your **tutor**.

Part B

Part B consists of **multiple** activities on LearningSpace. This work is **individual** and will be assessed on an **accumulated** basis.

Access the quizzes on LearningSpace. Answer all of the multiple-choice questions and give appropriate responses to the code writing exercises.

Your tutor will offer advice in the timetabled tutor meeting. You will receive immediate **automated feedback** from the LearningSpace.

Part C

Part C consists of a **single formative submission**. This work is **collaborative** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

- Several algorithms have been selected;
- Suggestions to improve maintainability are appropriate;
- Clear LICENSE and README present;
- Attempt to justify the software license is present;
- Evidence of collaborative programming in the repo.

To complete Part C, complete the online activities on the LearningSpace.

Firstly, review the pseudocode for the algorithms. You are expected to translate technical notation into executable code. As such, it is anticipated that you incorporate several of the listed algorithms into your solution. Clearly list these algorithms and state where they are implemented in the README.md file. You are welcome to use third-party libraries and resources in your solution, but these will not 'count' as translations for assessment purposes. Algorithms you draw from the literature *can* count if there is evidence of translation.

Then, complete the activities on LearningSpace. These will prompt you to outline how you will work together to improve maintainability, and to more clearly justify the terms of the software license. It is also worthwhile at this stage to review your git log to reflect upon your progress and the extent which you are using version control appropriately. Show these to your tutor in your timetabled meeting.

You will receive immediate **informal feedback** from your **tutor**.

Part D

Part D is a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

- Submission is timely;
- Enough work is available to conduct a meaningful review;
- A broadly appropriate review of a peer's work is submitted.

To complete Part D, prepare a draft version of the computer program. Ensure that the source code and related assets are pushed to version control and a copy is submitted to the peer-review workshop activity on LearningSpace prior to the scheduled peer-review session. Then, attend the scheduled peer-review session.

Please note you submit and review as individuals for the peer-review.

You will receive immediate **informal feedback** from your **peers**.

Part E

Part E is a **single summative submission** followed by a **demonstration**. The initial submission is **collaborative** whilst the demonstration is **individual**. These will be assessed, respectively, on a **criterion-referenced** basis and a **threshold** basis. Please also refer to the marking rubric at the end of this document for further detail. The following criteria are used to determine a pass or fail:

- Enough work is available to hold a meaningful discussion;
- Clear evidence of programming knowledge;
- No breaches of academic integrity.

To complete Part E, revise the computer program based on the feedback you have received. Then, push it into version control. As a reminder, formal submissions are through the LearningSpace, so please check the link to your repository on Falmouth's GitHub Enterprise server is valid.

Then, prepare a practical demonstration of your computer program. Ensure that the source code and related assets are pushed to version control and available to assessors prior to the scheduled viva session. Then, attend the scheduled viva session.

Please note you attend the viva as an individual.

There is an expectation that you work with other members of your group as

well as other students in your programming tutor group as part of your learning. You have permission to do this. However, you must *understand* the code in your submission and you must *acknowledge* the contributions of peers to your codebase explicitly and in accordance with academic conventions. As such, you should be able to explain **EACH** section of your solution individually, and place appropriate references in the comments of your source files. In the viva you will discuss the key programming constructs used in your solution, the key design decisions made when structuring the code, and how functionality was achieved. This is your opportunity to demonstrate your academic integrity and individual programming knowledge.

You will receive immediate **informal feedback** from your **tutor**.

You will receive **formal feedback** from your **tutor** up to three weeks after the final submission deadline.

Additional Guidance

It is critically important that you do not neglect your individual roles in the development process. Programming in pairs means that you work together on the same computer—switching between driver and navigator. It is a great opportunity to develop your technical communication skills and overcome common misconceptions about programming. It should not, however, be treated as a 'free ride'—you will get to review each others' progress.

You are not only being expected to generate new encodings from primitive input, but also *transform* and *repurpose* encodings that already exist. These could be melodies, sound effects, or other audio clips which you can load into your data structure. Please do load audio files. However, when using audio you have not authored yourself, the source should be noted in your README file and all relevant rights (e.g., copyright, license conditions, etc.) ahered to.

You can and should go beyond the techniques introduced in the lectures and the Guzdial book (e.g. researching algorithms for producing or manipulating graphics).

You are not being assessed on speed or memory performance. Do not worry too much about framerate, etc.

A common pitfall is poor planning or time management. Often, students underestimate how much work is involved in first learning programming concepts and then actually applying them. Programming is quite unlike other subjects in that it cannot be crammed into a last minute deluge just before a deadline. It is, therefore, very important that you begin work early and sustain a consistent pace: little and often.

It is very important to make a start on this project so you receive early feedback to give you some direction and to encourage you to practice your programming skills across the entire duration of the course. Ideally, you should be programming every day!

The peer-review component of this work does sometimes raise alarm. However, the only way to learn how to review code is by reviewing code. Your tutor will guide you through the process and provide advice. With practice, it will become clear what is satisfactory by discussing the quality of work with your peers and your tutor during the peer review sessions.

FAQ

- **What is the deadline for this assignment?**
  Falmouth University policy states that deadlines must only be specified

on the MyFalmouth system.

- **What should I do to seek help?**
  You can email your tutor for informal clarifications. For informal feedback, make a pull request on GitHub.
- **Is this a mistake?**
  If you have discovered an issue with the brief itself, the source files are available at:

  https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs
  .
  Please raise an issue and comment accordingly.

Additional Resources

Please review the LearningSpace for additional resources.

Marking Rubric

Criteria marked with a ‡ are shared by the group. All other criteria are individual.

To **pass** this assignment (achieve 40% or more), you must demonstrate adequate ability to generate ideas, problem solving, concepts, technical competency and proposals in response to the set brief. Your work must reflect an adequate, ethically informed, "real-world" experience as if you were in a industry/business environment or market. Enough of your work must be available to demonstrate your programming knowledge and allow a meaningful discussion to take place in the viva. You must also have satisfactory participation in all parts of the assignment and have submitted the final deliverable.

Criterion	Weight	Near Pass	Adequate	Competent	Very Good	Excellent	Outstanding
Basic Competency Threshold	30%	At least one part is missing, incomplete, or inadequate. Breach of academic integrity.	All quizzes completed. Peer review complete. Comments to peers are satisfactory, but may be minimal and/or basic.	As per quiz performance. Peer reviews are at least diagnostic and reassuring.	As per quiz performance. Peer reviews are instructive and encouraging, integrating appropriate insights.	As per quiz performance. Peer reviews are well-formed and diplomatic, conveying effective advice.	As per quiz performance. Peer reviews are attentive, apt, pithy, and constructive, conveying critical insight that aids improvement.
PROCESS: Functional Coherence of Code	5%‡	There is an attempt at implementing the algorithms, but without success. The source code does not compile, or might return erroneous results, but could work with minor changes.	At least one algorithm has been implemented successfully. There are many obvious logical errors, more than one of which is significant.	At least two algorithms have been implemented successfully. There are several obvious logical errors, no more than one of which is significant.	At least three algorithms have been implemented successfully. There are some obvious logical errors, which are not significant. The brief has been satisfied.	At least four algorithms have been implemented successfully. There are few obvious logical errors, which are cosmetic and/or superficial. The brief has been satisfied.	At least five algorithms have been implemented successfully. There are no obvious logical errors. The brief has been satisfied.
PROCESS: Sophistication of Code	15%‡	Insight into programming constructs is evident from the code. There is an attempt to structure the program (e.g. not just one monolithic function) but it doesn't include foundational elements.	Some insight into the appropriate use of programming constructs is evident from the code. All foundational elements of programming (e.g. representation, sequence, operations, branching, iteration, etc.) are present.	Much insight into the appropriate use of programming constructs is evident from the code. Appropriate use of functions with arguments supports a program structure, minimising redundancy.	Considerable insight into the appropriate use of programming constructs is evident from the code. Appropriate use of a rich mixture of structural elements (e.g. structs, enums, classes, etc.) supports a more extensible program structure.	Significant insight into the appropriate use of programming constructs is evident from the code. The program structure is effective, using data structures (e.g., *n*d arrays, hash tables, etc.) and constructs (e.g., delgates, generics, polymorphism, etc.) to better cohesion and coupling.	Extensive insight into the appropriate use of a wide repetoir of programming constructs is evident from the code. The program structure is highly effective, demonstrating high cohesion and low coupling.
PROCESS: Maintainability of Code	15% ‡	Insight into maintainability is evident from the source code. There is an attempt to make use of sensible variable names. There is an attempt to format and comment code.	The source code is commented sporadically. Many identifier names are clear and appropriate. Code formatting is mostly consistent.	Code is commented somewhat well. Most identifier names are descriptive and appropriate. The formatting style aids readability. There is little obvious duplication of code or of literal values.	Code is reasonably commented and doc-commented. Almost all identifier names are descriptive and appropriate. Almost all code adheres to a sensible formatting style, enhancing readability. There is almost no obvious duplication of code and constants are used appropriately.	Code is well commented and doc-commented. All identifier names are descriptive and appropriate. All code adheres to an appropriate formatting style. Liitle to no obvious duplication of code nor many unneeded literals. Some values can be easily "tinkered" but might still be in the code.	Code is very well commented and doc-commented. All identifier names are descriptive and appropriate. All code adheres to an appropriate formatting style. There is no obvious duplication of code nor any unneeded literals. All values are, where appropriate, easily "tinkered" outside of the code.

Marking Rubric

Criteria marked with a ‡ are shared by the group. All other criteria are individual.

To **pass** this assignment (achieve 40% or more), you must demonstrate adequate ability to generate ideas, problem solving, concepts, technical competency and proposals in response to the set brief. Your work must reflect an adequate, ethically informed, "real-world" experience as if you were in a industry/business environment or market. Enough of your work must be available to demonstrate your programming knowledge and allow a meaningful discussion to take place in the viva. You must also have satisfactory participation in all parts of the assignment and have submitted the final deliverable.

Criterion	Weight	Near Pass	Adequate	Competent	Very Good	Excellent	Outstanding
INDUSTRY: Creative Response to Brief	5%‡	There is something creative, but it is very modest. The work is a clone of an existing work with mere cosmetic alterations.	Little creativity. The work is derivative of existing works, with only minor alterations.	Some creativity. The work is derivative of existing works, demonstrating little divergent and/or subversive thinking.	Much creativity. The work is somewhat novel, demonstrating some divergent and/or subversive thinking.	Considerable creativity. The work is novel, demonstrating significant divergent and/or subversive thinking.	Significant creativity. The work is novel, with strong evidence of divergent and/or subversive thinking.
INDUSTRY: Ethically Informed	10%‡	There is a marginally appropriate license and/or at least partial compliance with intellectual property law.	There is a somewhat appropriate license. There is implicit recognition of intellectual property rights.	There is an appropriate license. There is explicit recognition of intellectual property rights in the README.md. Acknowledgements are clearly demarcated in the source code.	There is a suitable license. There is explicit description of intellectual property rights in the README.md. Authorship is demarcated in the source code. Copyright notices are present.	There chosen license is suitable. There is explicit explanation of intellectual property rights in the README.md. Authorship is accurately declared in the header using appropriate standards and demarcated in the source code. Copyright notices are present and appropriate.	There chosen license is suitable. There is explicit justification of intellectual property rights in the README.md. Authorship is accurately declared in the header using appropriate standards and demarcated in the source code. Copyright notices are accurately declared in the header using appropriate standards. There may be reference to a transfer agreement.
INDUSTRY: Use of Version Control	20%	Version control attempted, but not properly used.	Source code has seldom been checked into version control.	Source code has been checked into version control at least once per week. Sensible commit messages are present.	Source code has been checked into version control several times per week. Commit messages are clear, concise and relevant.	Source code has been checked into version control many times per week. Commit messages are clear, concise and relevant.	Source code has been checked into version control with considerable frequency. Commit messages are clear, concise and relevant.