TINKERING GRAPHICS

Version 3.0
Computing
COMP120

Dr Michael Scott

Introduction

In this assignment, you are required to write a computer program that will *tinker* with an existing digital graphic in a creative way.

"For every complex problem there is an answer that is clear, simple, and wrong."

— Henry Mencken

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. You will, therefore, leverage the principles you have learned to exercise your creativity through computer software. This will prepare you to tackle challenges in many creative domains.

This assignment is formed of several parts:

"Bad programming is easy. (People) can learn it in 21 days, even if they are dummies... (Good programming requires a) willingness to devote a large portion of one's life to deliberative practice... So go ahead, buy that book; you'll probably get some use out of it. But you won't change your life or your real expertise as a programmer in 21 days... How about working hard to continually improve over 24 months? Well, now you're starting to get somewhere..."

— Peter Norvig

(A) **Select**, as a **pair**, **ONE** of the contracts provided by your tutor and:
  i. **state** which contract you will work on;
  ii. **list** the requirements implied by the contract;
  iii. **declare** the terms of your software license;

(B) **Write**, as a **pair**, a draft computer program that will:
  i. **address** the requirements implied by the contract and relevant intellectual property law;
  ii. **implement** at least **FOUR** algorithms for tinkering graphics;

(C) **Write**, as a **pair**, a final computer program that will:
  i. **revise** any issues raised by your tutor and/or your peers.

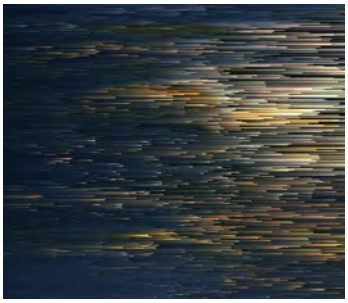
(D) **Present**, as an **individual**, a practical demo of the computer program to your tutor that will:
  i. **demonstrate** your academic integrity;
  ii. as well as **demonstrate** your **individual** programming knowledge.

Assignment Setup

This assignment is a **pair programming task**. Fork the repository at:

https://gamesgit.falmouth.ac.uk/scm/comp120/comp120-tinkering-graphics.git

Extend the repository with sub-directories as appropriate. Ensure that you maintain the README.md and LICENSE.md files. Modify the .gitignore to the defaults for **C#** and your development environment of choice.



@pixelsorter is a Twitter bot written in Ruby that sorts the rows (or optionally columns) of an image according to a specific method like hue, red, brightness, luma, etc.

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, write about your contract in the relevant LearningSpace activity. Show this to your tutor in your timetabled meeting. You are to select **ONE** contract to tackle together with your partner and to suggest an appropriate license for your code. If acceptable, this will be signed-off.

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

Part B

Part B is 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:

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

To complete Part B, prepare a draft version of the computer program. Ensure that the source code and related assets are pushed to version control and a link is submitted to the peer-review workshop activity on LearningSpace prior to the scheduled peer-review session. You may have to set permissions on your repository to allow other students to review your code. Then, attend the scheduled peer-review session.

You are expected to translate technical notation into executable code. As such, it is anticipated that you incorporate **FOUR** 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.

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

Part C

Part C is a **single summative submission**. This work is **collaborative** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of this document for further detail.

To complete Part C, revise the computer program based on the feedback you have received. Then, push it into version control. Formal submissions are through the LearningSpace, in this case a link to your repository on GamesGIT. Only one of you needs to submit the link to the repository on LearningSpace.

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

Part D

Part D is a **single summative submission**. This work is **individual** and will be assessed on a **threshold** basis. 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 D, 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.

There is a clear expectation that you pair program with your partner. As such, you should **EACH** be able to explain the solution individually. 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**.

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 being expected to *transform* and *repurpose* encodings (i.e. manipulating existing pictures). However, you may create your own images if desired. When using images you have not authored yourself, the source should be noted in the README.md file and all relevant rights (e.g., copyright) acknowledged.

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.

The first deadline is quite close to the start of the course and not much material will have been covered by this point. Please rest assured. This first formative submission is supposed to be a simple analysis of requirements. We expect there to be errors. However, 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.
- **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

All submissions and assessment criteria for this assignment 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.					
PROCESS: Functional Coherence of Code	5%	There is a reasonable attempt at implementing the algorithms. The source code does not compile, or might return erroneous results, but would 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 four 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 source code. There is an attempt to structure the program (e.g. not just one monolithic function) but it doesn't include key elements (e.g. functions with arguments).	Some insight into the appropriate use of programming constructs is evident from the source code. All key elements of program structure (e.g. functions with arguments) are present.	Much insight into the appropriate use of programming constructs is evident from the source 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 source code. Appropriate use of a rich mixture of structural elements (e.g. classes) supports an extensible program structure.	Significant insight into the appropriate use of programming constructs is evident from the source code. The program structure is effective, having a reasonable level of cohesion and coupling.	Extensive insight into the appropriate use of programming constructs is evident from the source code. The program structure is very effective, demonstrating high cohesion and low coupling.
PROCESS: Maintainability of Code	15%	Insight into maintainability is evident from the source code, with at least one comment. There is an attempt to make use of sensible variable names. There is an attempt to format code.	The source code is commented sporadically. Many identifier names are clear and appropriate. Code formatting is mostly consistent, but might not aid readability as well as it could.	Source code is somewhat well commented. Most identifier names are descriptive and appropriate. An attempt has been made to adhere to an appropriate formatting style. There is little obvious duplication of code or of literal values.	Source code is reasonably well commented. Almost all identifier names are descriptive and appropriate. Almost all code adheres to an appropriate formatting style. There is almost no obvious duplication of code or of literal values.	Source code is reasonably well commented, with doc-comments. All identifier names are descriptive and appropriate. All code adheres to an appropriate formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered" but might still be in the source code.	Source code is very well commented, with doc-comments. All identifier names are descriptive and appropriate. All source code adheres to an appropriate formatting style. There is no obvious duplication of code or of literal values. Most literal values are, where appropriate, easily "tinkered" outside of the source code.

Marking Rubric

All submissions and assessment criteria for this assignment 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	10%	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	15%	Version control (i.e. GamesGit) has been used only once or twice.	Source code has seldom been checked into version control (i.e. GamesGit).	Source code has been checked into version control (i.e. GamesGit) at least once per week. Sensible commit messages are present.	Source code has been checked into version control (i.e. GamesGit) several times per week. Commit messages are clear, concise and relevant. There is evidence of somewhat meaningful engagement with peers (e.g. code review). Comments to peers are somewhat constructive and provide some insight.	Source code has been checked into version control (i.e. GamesGit) several times per week. Commit messages are clear, concise and relevant. There is evidence of meaningful engagement with peers (e.g. code review). Comments to peers are reasonably constructive and provide much insight.	Source code has been checked into version control (i.e. GamesGit) many times per week. Commit messages are clear, concise and relevant. There is evidence of effective engagement with peers (e.g. code review). Comments to peers are reasonably constructive and provide considerable insight.