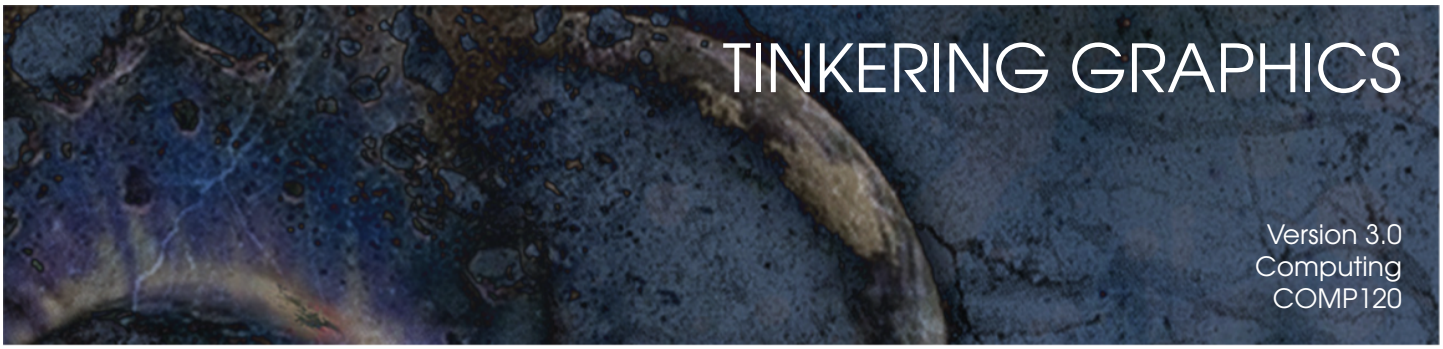


TINKERING GRAPHICS

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 computer graphic in a creative way.

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.

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

— Henry Mencken

This assignment is formed of several parts:

(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 in Python that will:
  i. **address** the requirements implied by the contract and relevant intellectual property law;
  ii. **implement three** algorithms for tinkering graphics;

(C) **Write**, as an **individual**, a final computer program in Python 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.

"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

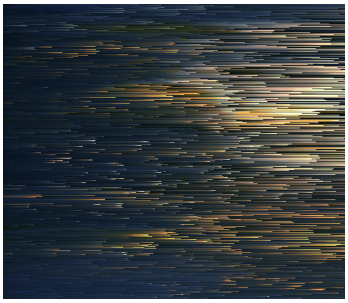
Assignment Setup

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

https://github.com/Falmouth-Games-Academy/comp120-tinkering-graphics

Use the existing directory structure and, as required, extend this structure with sub-directories. Ensure that you maintain the readme.md file.

Modify the .gitignore to the defaults for **Python**. Please, also ensure that you add editor-specific files and folders to .gitignore.



@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 readme.md document. Show this to your tutor in-class. 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 draft versions of the computer programs. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled peer-review session. Then, attend the scheduled peer-review session.

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, upload it to the LearningSpace. Please note, the LearningSpace will only accept a single .zip file.

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 the computer programs. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled viva session. Then, attend the scheduled viva session.

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 GitHub 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. 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

- Guzdial, M.J . and Ericson, B. (2015) Introduction to Computing and Programming in Python: A Multimedia Approach, 4th Edition. Pearson: New York.
- Martin, R.C. (2008) Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall: New York
- http://guide.agilealliance.org/guide/pairing.html
- http://www.pairprogramming.co.uk/
- http://www.pythontutor.com/

Marking Rubric

All submissions and assessment criteria for this assignment are individual.

Criterion	Weight	Near Pass	Adequate	Competent	Very Good	Excellent	Outstanding
Basic Competency Threshold	30%	At least one part is missing or is inadequate. Breach of academic integrity.	Adequate ability to generate ideas, problem solving, concepts, technical competency and proposals in response to set briefs and/or self-initiated activity.	The work demonstrates an adequate, ethically informed, real-world experience of industry/business environments and markets.	Enough work is available to hold a meaningful discussion.	Adequate participation in-class peer-review activities at least at the level of basic competency.	Clear evidence of programming knowledge.
PROCESS: Functional Coherence of Code	5%	No algorithm has been implemented successfully. The source code does not compile or there are serious syntax errors.	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 three algorithms have been implemented successfully. There are few obvious logical errors, which are cosmetic and/or superficial. The brief has been satisfied.	At least three algorithms have been implemented successfully. There are no obvious logical errors. The brief has been satisfied.
PROCESS: Sophistication of Code	15%	No insight into the appropriate use of programming constructs is evident from the source code. No attempt to structure the program (e.g. one monolithic function).	Little insight into the appropriate use of programming constructs is evident from the source code. The program structure is poor.	Some insight into the appropriate use of programming constructs is evident from the source code. The program structure is adequate.	Much insight into the appropriate use of programming constructs is evident from the source code. The program structure is appropriate.	Considerable insight into the appropriate use of programming constructs is evident from the source code. The program structure is effective. There is high cohesion and low coupling.	Significant insight into the appropriate use of programming constructs is evident from the source code. The program structure is very effective. There is high cohesion and low coupling.
PROCESS: Maintainability of Code	15%	There are no comments in the source code, or comments are misleading. Most variable names are unclear or inappropriate. Code formatting hinders readability.	The source code is only sporadically commented, or comments are unclear. Some identifier names are unclear or inappropriate. Code formatting is inconsistent or does not aid readability.	The source code is somewhat well commented. Some identifier names are descriptive and appropriate. An attempt has been made to adhere to thhe PEP-8 formatting style. There is little obvious duplication of code or of literal values.	The source code is reasonably well commented. Most identifier names are descriptive and appropriate. Most code adheres to the PEP-8 formatting style. There is almost no obvious duplication of code or of literal values.	The source code is reasonably well commented, with Python doc-strings. Almost all identifier names are descriptive and appropriate. Almost all code adheres to the PEP-8 formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered" in the source code.	The source code is very well commented, with Python doc-strings. All identifier names are descriptive and appropriate. All source code adheres to the PEP-8 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.
INDUSTRY: Creative Response to Brief	10%	No creativity. 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 highly original, with strong evidence of divergent and/or subversive thinking.

Marking Rubric

All submissions and assessment criteria for this assignment are individual.

Criterion	Weight	Near Pass	Adequate	Competent	Very Good	Excellent	Outstanding
INDUSTRY: Ethically Informed	10%	There is an inappropriate license and/or misuse of intellectual property.	There is an minimally 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 (e.g. PEP257) 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 (e.g. PEP257) and demarcated in the source code. Copyright notices are accurately declared in the header using appropriate standards (e.g. module-level dunder names, PEP8). There may be reference to a transfer agreement.
INDUSTRY: Use of Version Control	15%	Version control (e.g. GitHub) has not been used.	Source code has been checked into version control (e.g. GitHub).	Source code has been checked into version control (e.g. GitHub) at least once per week. Sensible commit messages are present.	Source code has been checked into version control (e.g. GitHub) 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 (e.g. GitHub) 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 (e.g. GitHub) 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.