Bournemouth University Faculty of Media and Communication	**Coursework Assignment Brief**	2019/20
	Programme Title: BA (Hons) Computer Animation Technical Arts	Level 4
	Unit Title: Programming Principles	
	Title of Brief: Programming Project	This assignment is a formal element of coursework worth 50% of the overall unit mark (Each piece of coursework may vary according to the unit)

THE BRIEF

This assignment requires the design and implementation of a computer graphics application supported by a report discussing its design and implementation (counting for 50% of unit mark). It assesses software development practice, knowledge of procedural programming concepts as well as understanding of basic 2D computer graphics techniques and algorithms. This is in relation to the following ILO's (Intended Learning Outcomes) of the Programming Principles Unit:

- Being able to implement algorithms in general and basic CG algorithms using a suitable Application Programming Interface (API).
- Demonstrating an ability to design and implement suitable data structures.
- Demonstrating an ability to apply suitable software engineering principles.

All assignments must be written in C (not C++) and make use of the SDL library (and appropriate SDL extensions such as SDL_image or SDL_ttf) for creating windows, accessing images and pixel drawing operations (as well as OpenGL for option 3) and must include a working "makefile" that will successfully build the project on the lab machines (on Linux).

Project Option 1: Turtle Graphics & Drawing Spirolateral Curves

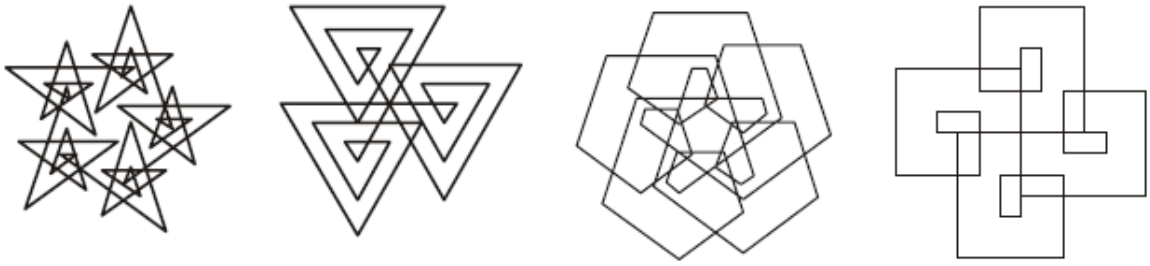
Turtle graphics (http://en.wikipedia.org/wiki/Turtle_graphics) is a simple and well know method for drawing complex shapes from simple lines. Turtle graphics employs the concept of a "**turtle**", which is a simple robot or agent in the shape of a turtle that has the following attributes (usually implemented using a record data structure):

- a **position** (x, y) in 2D space,
- a **heading** or **angle** measured in degrees or radians, and
- a flag/variable **pen** that could be set either as Up (True/drawing) or as Down (False/not drawing)

If the **turtle's pen** is set to have the value Down and the **turtle** moves then the **turtle** leaves a trace while it moves. It accepts the following simple instructions (usually implemented as functions):

- **pen_up** sets the state of the turtle's **pen** to Up,
- **pen_down** sets the state of the turtle's **pen** to Down,
- **move n** makes the turtle move **n** units along its **heading** (usually implemented by drawing a line segment between the start point and the end point), and
- **turn α** turns the **heading** of the turtle by **α** degrees.

While there are some similarities between turtle graphics and L-systems, turtles and L-systems are not the same and need to be treated/implemented differently.



A spirolateral is a geometric shape/pattern that is constructed from a series of straight lines of growing

length that are attached at an angle, which, if repeated after one another, will result in a closed curve (http://mathworld.wolfram.com/Spirolateral.html).

Your task is to first implement a simple **turtle library** (data structure & a set of related functions – not implemented as an L-system) using **C** in combination with the **SDL** library for 2D pixel drawing and then to use this library to create a spirolateral drawing program with a simple user interface (that may be graphical or text-based) that is capable of the following:

1. allow the user to interactively (through the user interface) select/change the drawing parameters such as drawing and background colours, the initial length and number of line segments. These parameters should also include the number of repetitions of the initial pattern, the drawing angle etc.
2. provide a set of (at least) three pre-set spirolaterals that are selectable by the user for drawing
3. save the resulting spirolaterals as image files, and optionally save the spirolateral drawing as an image sequence that shows the addition of each line segment as a separate (animation) frame that would show the curve being drawn step by step

Recommended reading:

- Krawczyk, R.J. (1999). "Spirolaterals, Complexity from Simplicity", in Proceedings of the 1999 Conference of The International Society of the Arts, Mathematics and Architecture. Available from: http://mypages.iit.edu/~krawczyk/isama99.pdf
- Krawczyk, R.J. (2000). "The Art of Spirolaterals", in Proceedings MOSAIC 2000, the Millennial Open Symposium on the Arts and Interdisiciplinary Computing. Available from: http://mypages.iit.edu/~krawczyk/mosaic00.pdf
- Reed, D. (2002), "CSC 107, Lab 2: Functions and Turtle Graphics", in Nifty Assignments – SIGCSE 2002. Available from: http://dave-reed.com/Nifty/WalkLab.html

Project Option 2: Advanced Etch-A-Sketch

Your task is to write a computer graphics program in **C** (and using the **SDL** library) that allows image creation by emulating the popular Etch-A-Sketch toy (https://en.wikipedia.org/wiki/Etch_A_Sketch), adding a set of advanced features.

Unlike the original Etch-A-Sketch toy, your program should allow the creation of multi-coloured images. The program should be able to save resulting images and to reload previously saved images to continue drawing. The left and right rotating control buttons should be emulated by using the 'A' & 'D' and the left-arrow & right-arrow keys.

As a minimum, the program should include the following features:

1. provide a keyboard (shortcut) based interactive user interface
2. allow the user to change drawing colour and line-width for drawing
3. allow the user to save the resulting image in an image file of their choosing and take a snapshot of current drawing position and parameters (a 2nd file – either a text file or a binary file storing data in a format of your choosing – noting the drawing colour, drawing position, line width and the corresponding image file will have to be created for this) and to reload previously saved images

for further editing

Recommended reading:

- http://gigi.nullneuron.net/gigilabs/sdl2-pixel-drawing/
- https://wiki.libsdl.org/CategoryKeyboard
- https://www.programiz.com/c-programming/c-file-input-output

Project Option 3: Breakout

Your task is to write a clone of the classic arcade video game "Breakout" (released in 1976), making use of the **OpenGL API** in conjunction with the **SDL** library (see: https://en.wikipedia.org/wiki/Breakout_(video_game), https://en.wikipedia.org/wiki/Arkanoid and https://www.youtube.com/watch?v=Up-a5x3coC0).

In the game Breakout, the player controls a horizontal paddle at the bottom of the screen that is used to hit a bouncing ball towards a wall of bricks at the top of the screen, which get 'destroyed' when hit with the ball (different types of bricks may need a different number of hits to break). The speed of the ball is variable, depending on different factors (e.g. the 'angle' of connection with the paddle) If the player misses the ball, i.e. if the ball falls below the bottom of the screen, the player loses a 'life'.

As a minimum, your program should include the following features:

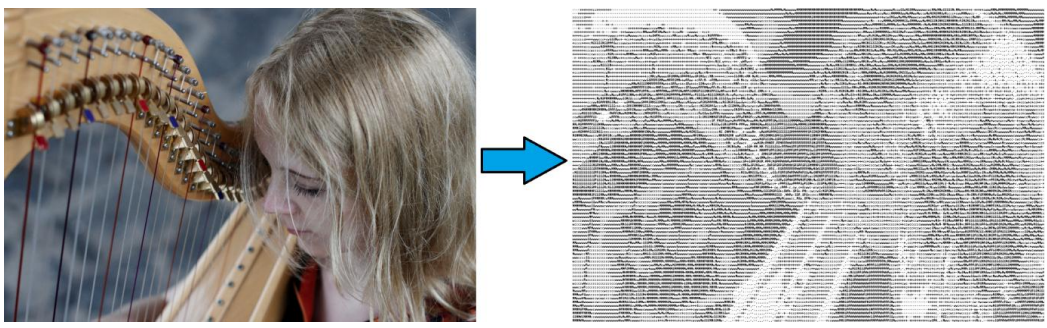
1. a keyboard-controlled paddle for the player and three player 'lives'
2. two different types of brick and three different levels that show a different target-wall (hard-coded or possibly loaded from a file)
3. at least one power-up (e.g. gained by hitting a certain number of bricks in a short time or a special brick being destroyed etc.) that provides a bonus to the player (e.g. short-term slow-down of the ball, a wider paddle, an extra 'life' etc.)

Recommended reading:

- https://www.c64-wiki.com/wiki/Arkanoid

Project Option 4: Frame Processor

Your task is to write a computer program in **C** (making use of the **SDL** library) that converts source image(s) to ASCII art (see: http://en.wikipedia.org/wiki/ASCII_art), i.e. a representation of the original image using ASCII characters only.



Employing a text-based interface (either through command-line arguments or through interactive queries during program execution), the program you create should allow the user to select an image or a sequence of images (e.g. animation frames) and then process these (by matching pixel regions with ASCII characters of comparable intensity/brightness), saving the results in a format selectable by the user, which should be either as ASCII text files, an image file of ASCII characters (with a selectable text and background colour) or as an image of coloured ASCII characters (using the average colour of the pixels they replace) on a neutral background (using the SDL_ttf extension for the SDL library to write characters into the image).

Do not hard-coded filenames of files that the user is supposed to manipulate (e.g. the filename for images a user loads or saves)!

Recommended reading:

- Parberry, I. (2011). Text Art - http://ianparberry.com/art/ascii/text/
- Parberry, I. (2011). ASCII Art on a Pixel Shader – http://ianparberry.com/art/ascii/shader/
- Mikolay, M. (2012). A Basic ASCII Art Algorithm – https://web.archive.org/web/20151031150811/http://mattmik.com/articles/ascii/ascii.html

Marking Criteria:

- The quality of the source code and usability of the program. This includes effective use of programming techniques, such as using structured and procedural programming, recursion, relevant control structures and data structures, etc., but also encompasses adherence to the brief (such as the use of the prescribed programming language and libraries). This will count for 50% of the mark of your assignment.

The remaining 50% will be determined by:

- The project report, including Background, Implementation, Results (see section 3 of the submission details below) and references using BU Harvard referencing.
- The (visual) Impact (i.e. how well it achieves the intended results) of the generated artefacts (based on the Results section of your report, and/or submitted generated artefacts, see submission details).
- Source code documentation (relevant and appropriate comments in the source code).

SUBMISSION DETAILS

Your submission needs to include the following:

1. One or more well commented C source code files **and** a working **Makefile** that will build an executable program without errors on any machine (on Linux) in the 1st year labs and that will run on the machines in the 1st year labs.
2. A PDF user Manual for your application. The User Manual should explain how to initialise and run your program (this does not include compilation).
3. A **PDF report** documenting and explaining your application. As a guide, the report should be approximately 6-8 A4 pages of text and appropriate diagrams (such as design drawings/flow charts but also including resulting images/screenshots). The report should contain a section called “Background” or “Introduction” explaining related work, the algorithms, techniques, and ideas used in your project. It should also have a section called “Implementation” explaining the structure of your program in terms of the implementation of the algorithms and techniques used, describing flow of control, and explain the implementation of the most important functions or procedures. This section should serve to illuminate but not replicate your code. Finally, the report should contain a section called “Results” which includes images, or references to images (and video) demonstrating and explaining the results produced by your program. Sources used or referred to must be cited using BU Harvard referencing.
4. Appropriate sample results (and source data, such as original and images where this applies), such as animations or images generated by your program.

Note: this assignment is not a group project assignment but an individual project assignment,

and apart from trivial parts of your source code, we expect to see a substantial amount of original/different source code and original reports submitted by different students.

Ensure all requested deliverables are present. **Make sure your programs compile and run on the Linux workstations in the lab**. It is your responsibility to include examples, images (screen shots), other data and relevant documentation regarding the operation of the programs submitted.

Make sure that any 3rd party source code (e.g. found in a book or downloaded from the web) or source code you may have been given by fellow students or members staff is credited, i.e. cited in the report and included in comments explaining where the code was found and stating who is the author – if you received help with parts of your code the person helping you should be acknowledged in a source code comment as well as in your report. Borrowed/3rd party code should be bracketed by comments:

```
/* source from "Author" starts here; it originates from ... and does ... */
… code …
/* source from "Author" ends here */
```

If the author is unknown state the web-site or location where the source code was found. If the source is modified, but the modifications are small, start and end the comments with " `source based on "Author"` ". For example :

```
/* source from Ari Sarafopoulos starts here : ~/asarafop/CFG */
int ival(double v) /* return the closest integer */
{
   return ((v - floor(v)) > 0.5) ? (int)ceil(v): (int)floor(v);
}
/* source from Ari Sarafopoulos ends here */
```

Submit via the assignment submission link in the navigation bar in the Programming Principles unit on Brightspace.
Please be aware of the following restrictions.

- Individual files (including zipped folders) must be less than **3GB** (*considering previous cohorts' submissions, your project files should not exceed 50MB*)**.**
- It is strongly recommended that you upload large files while on campus using a University computer.

It is your responsibility to ensure that you are submitting to the correct submission box in the correct unit on Brightspace. You should check and retain your receipt for all submissions. If you are uploading multiple files you should check all files are listed on your receipt.

DEADLINE

12 Noon, 20th January 2020 is the submission deadline for this assessment, using the Brightspace on-line submission system – please note that this is the time by when your submission must have finished uploading. If you encounter a technical (Brightspace) problem with uploading your submission you must contact the IT Service Desk to log the problem **immediately and at least 15 minutes before the deadline** (phone 65515 if calling internally or 01202 965515 if calling externally and log an IT job using the Self Service web tracker http://servicedesk.bournemouth.ac.uk).

Ensure all requested deliverables are present. Make sure your programs compile (using the makefile) and run on the Linux workstations. It is your responsibility to include examples, images (screen shots), other data and relevant documentation regarding the operation of the programs submitted.

Please note that this (**12 Noon, 20th January 2020**) is the final time you can submit – not the time to submit! Your feedback and mark for this assignment will be provided on the 10th of February 2020 online (on Brightspace). If you fail your assignment or need further clarification on your feedback please make an appointment to see your tutor.

HELP AND SUPPORT

If you have any questions, please contact Dr Eike Anderson either in the labs or by email: EAnderson@bournemouth.ac.uk;
For additional on-site help and support, you can also contact the animation demonstrators.

- Plagiarism and Self-Plagiarism: You must acknowledge your source every time you refer to others' work or work that you have previously submitted and been assessed on, using the **Harvard Referencing** system (Author/Date method). Failure to do so amounts to plagiarism or self-plagiarism which is against University regulations.

  Please refer to http://libguides.bournemouth.ac.uk/study-skills-referencing-plagiarism for further details of this and to https://www1.bournemouth.ac.uk/discover/library/using-library/how-guides/how-cite-references for the University's guide to citation in the Harvard style.

  Students must ensure that they do not commit any type of Academic Offence. For further information please see: https://www1.bournemouth.ac.uk/discover/library/using-library/how-guides/how-avoid-academic-offences
- Plagiarism regulations extend to audio visual materials and work in other media. Archive or other material not generated by yourself or crew must be appropriately captioned when on screen and an acknowledgement to the source of the material included in the end credits or equivalent part of any online material. Failure to do so amounts to plagiarism or self-plagiarism, which is against University regulations.
- Students with **Additional Learning Needs** may contact Learning Support on http://studentportal.bournemouth.ac.uk/learning/als/index.html
- General **academic support** is available via the Study Skills community on Brightspace.
- **Additional support** for Faculty of Media and Communication students only is provided by a small team of Learning Development Tutors. Please contact FMCLearningDevelopment@bournemouth.ac.uk to make an appointment.
- If you have any valid **exceptional circumstances** that mean you cannot meet an assignment submission deadline and you wish to request an extension, you will need to complete and submit the Exceptional Circumstances Form for consideration to your Administrator together with appropriate supporting evidence (e.g., GP note) normally before the coursework deadline. Further details on the procedure and the exceptional circumstances form can be found at https://www.bournemouth.ac.uk/students/help-advice/looking-support/exceptional-circumstances *Please make sure you read these documents carefully before submitting anything for consideration.*
- Please be mindful that certain actions carry risk which should be adequately assessed before undertaking the activity. Please refer to University/Faculty guidance for clarification.

Disclaimer: The information provided in this assignment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and via the VLE and a new version of this assignment brief will be circulated.	Version: 1 (Updated September 2019)