Pixel Editor Design Report – Group 82

Jack Tudbury - 39102459, Sufyan Osman – 38947099, Nat Fu 39002519,
Jack Rawson - 38916894, Crystal Leong - 39217531, and Alisha Naseem – 39205592

Reason for Selected Topic

The Pixel editor was chosen due to the versatility of features, and group members familiarity of the use case. The scope that the Pixel editor offers is greater than that of the Vector editor and will allow for us to be more creative in what we develop. This will help us stand out and create a unique piece of software that's packed with extra features.

Software Requirements

Non-Functional Requirements:

- NF1: The system should load files in less than 1 second.
- NF2: The system should export all supported file types in less than 3 seconds.
- NF3: The software shall run on both Windows 10 (or later), and Unix-based systems without change in behaviour or performance.
- NF4: The system should be accessible to all.
- NF5: The system shall support images up to 1000x1000 pixels.
- NF6: The system shall provide keyboard shortcuts.
- NF7: The system shall use the Java Swing library.
- NF8: The system shall be developed using JDK 17.

Functional Requirements:

- F1: The system shall allow the user to import and export of files.
    - F1.1: Reloading specific files back into pixel editor.
    - F1.2: Enable users to drag and drop an image into the canvas.
- F2: The system shall allow for changes to be redone/undone.
- F3: The system shall be usable with a screen reader for accessibility
- F4: The system shall have a simple toolbar with core functions
    - F4.1: Layer selector/creator
    - F4.2: Canvas orientation (flip/rotate)
    - F4.3: Canvas cropping
    - F4.4: Pixel selector
    - F4.5: Brush tool
    - F4.6: Eraser tool
- F5: The system shall support varying colour identification schemes (e.g.: RGB).
- F6: The system shall support basic canvas and drawing tools
    - F6.1: Varying brush sizes.
    - F6.2: Bucket fill
    - F6.3: Colour picker
    - F6.4: Colour palette (loading/saving colours with a preset/saved colour palette)
    - F6.5: Blur Tool
- F7: The system shall implement a layer editing system
    - F7.1: The system shall allow for the addition and deletion of layers
    - F7.2: The system shall have a visibility toggle for each layer

    - F7.3: The system shall allow for a lock function where editing is not possible when activated
    - F7.4: The system shall allow for the grouping of layers
- F8: The system shall have a toggle-activated pixel grid
- F9: The system shall allow for zoom-in and zoom-out between 25% to 500%
- F10: The system should allow the user to edit PNGs compatible with Minecraft
    - F10.1: Minecraft skins
    - F10.2: Minecraft block textures
- F11: The system should show a preview of the user's edits on the appropriate 3D model
    - F11.1: The system should update the 3D model automatically

Features List

Name	Type	Description
File Management		
Save/Export	Files	With exporting images, PNG is preferred for reuse since it is lossless.
Import	Files	Allow importing of old file to be used again
Main Editor		
Colour Picker	Tool	Allows custom colour based on RGB values
Colour Palette	Tool	Provides a default colour palette and allows user to create custom palettes, including a recently used palette.
Eraser	Tool	Reverts the selected pixels to white
Select Pixels	Tool	Allow selection of various pixels to apply actions to them. Square selection will be easiest. Also allow the selection of all pixels which are the same colour.
Fill	Tool	Fills connected pixels that are the same colour along an axis
Blur	Tool	Changes the selected pixels to be a colour mixture of the pixels around them (creating a blurring effect)
Layers	Tool	Supports transparency, visibility toggling
Flip Image	Tool	Mirrors image
Rotate Image	Tool	Allows the rotations of the canvas in 90-degree increments
Varying Pen Sizes	Tool	Allows selection of different brush sizes
Undo/Redo	UI	Store the difference between the 2 images delta's rather than the images themselves for undo/redo.
Zoom	UI	Allows zooming in and out (25% to 500%)
Minecraft Skin Maker / Extra Features		
Minecraft Skin Maker	Extra Feature	Both for skin and block textures.
Preview	Extra Feature	Ability to show a preview of the user's edits on the appropriate 3D model on both skins and blocks

UI Mock-Up

Main Menu - When opening the program, the user will see an option to create a new file (either a new Minecraft skin, new pixel art or new Minecraft block) and an option to load an existing file. The create new file section will feature large buttons displaying the options and the loading a file section will be comprised of 2 lists. One list shows the name of the file, and the other list will be adjacent and display the type of file it is. At the top of this screen, and the others, will be minimise, maximise and closing options as well as the title of the software. (See Appendix 1.A)

Pixel Art Editor - The pixel art editor will be mostly taken up by the canvas. On the side of the screen will be a tool bar comprising of the paint brush, pencil, eraser, fill tool etc. This is also present for the skin and block editors. Below these tools are the colour selectors along with a small grid of recently used colours for quick access. Below these are the undo and redo buttons. The canvas is made up of pixels that are highlighted along their borders in the colour selected when the cursor hovers over them. (See Appendix 1.B)

Minecraft skin editor – This will be like the pixel art editor with the change of the right-hand side of the screen being taken up by a body part selector. It will feature a person that can have their body parts (head, torso, arms etc.) selected for editing. This helps create the Minecraft skin. Above the person will be a small menu to decide which side of the body part is to be edited. Additionally, on this side of the screen there will be buttons to preview and export the Minecraft skin. (See Appendix 1.C)

Minecraft Block editor – This will be very similar to the skin editor, the canvas will take up the left-hand side and the menu to select which side of the block to edit will be on the right-hand side. Additionally, for the block and skin editors, the selection menus will feature previews of what has already been drawn on the body parts and sides. (See Appendix 1.D)

Code Design

Our editor will be built using a modular object-orientated design using the model view controller (MVC) pattern to ensure there's a separation of concerns to make development easier. Additionally, it also makes data ownership clear and ensures future extensibility.

1. **High-level Overview of MVC Layers**

   The application is structured into three layers:

   Model (Data + Logic)

   - Manages core application logic
   - Responsible for:
     - Storing and maintaining canvas state – pixels, layers, colours, tools.
     - Handling image processing operations
     - Undo/redo functionality (editing history)
   - Ownership: Owns all the raw data (pixels, layers, colours) and ensures consistency across the system.

Model	`CanvasModel.java`
	`LayerModel.java`
	`ToolModel.java`
	`ColourPalette.java`
	`UndoManager.java`

View (UI)

- Manages the UI using Java Swing.
- Responsible for:
  - Displays canvas, toolbars, colour palettes, layers.
  - Updates display based on **Model** changes
  - Ensures the system is accessible and user-friendly.
- Ownership: Owns the UI elements but doesn't manipulate logic or handle pixel data directly.

View	`MainWindow.java`
	`CanvasPanel.java`
	`ToolBarPanel.java`
	`ColourPickerPanel.java`
	`LayerPanel.java`
	`MinecraftOverlay.java`

Controller

- Acts as the bridge between the **Model** and **View**.
- Responsible for:
  - Processing mouse clicks, keyboard shortcuts and file operations.
  - Changes the **Model** based on the user input.
  - Refreshes the **View** once the **Model** changes
- Ownership: Manages the user interactions and synchronises the state of the application.

Controller	`EditorController.java`
	`ToolController.java`
	`FileController.java`

2. Code Modules Explained

Below is an overview of the core classes and interfaces we plan to implement. The application design is guided by Model View Controller principles. Note that the actual code may not explicitly label classes with these terms. Instead, the responsibility of each class will be implemented in a way which best suit our projects while still ensuring a clear separation of concerns.

Class	Info
CanvasModel.java	**Purpose:** Stores pixel data for entire canvas. **Data Structure:** A list of `LayerModel` objects combined
LayerModel.java	**Purpose:** Represents a single editable layer in the editor which will hold pixel data. **Data Structure:** Could hold its own BufferedImage or partial pixel array.

ToolModel.java (Interface or Abstract Class)	**Purpose:** Defines a standard interface for all drawing tools to ensure each tool fulfils its required function.
	Methods:
	`onMouseDown(Point p, CanvasModel model)`
	`onMouseDrag(Point p, CanvasModel model)`

	`onMouseUp(Point p, CanvasModel model)` **Concrete Implementations:** `BrushTool, EraserTool, FillTool, BlurTool.`
ColourPalette.java	**Purpose:** Manages and stores colours defined by system or user including recently used. **Data Structure:** `ArrayList<Colour>`
UndoManager.java	**Purpose:** Implement undo/redo stack. **Data Structure:** Uses two stacks (one for undo and one for redo) that store a state change
MainWindow.java	**Purpose:** The main `JFrame` which contains the entire UI. **Components Contained:** - `CanvasPanel` - `ToolBarPanel` - `MenuBar` (for file operations, prefrences) - `StatusBarPanel` (show coordinates, zoom etc)
CanvasPanel.java	**Purpose:** Visual drawing area where the user interacts with the canvas. **Implementation:** - Custom swing component. - Listens for mouse events and notifies controller.
ToolBarPanel.java	**Purpose:** A tool selection interface. **Implementation:** - Use `JButtons` to represent each tool. - Notify ToolController when a tool is selected.
ColourPickerPanel.java	**Purpose:** Displays a colour palette with RGB sliders and Hex Code with a colour preview **Implementation:** Can be implemented using standard Swing components allowing users to select and modify colours.
LayerPanel.java	**Purpose:** Represents a single layer in the editor which will hold pixel data. **Implementation:** List of all layers with controls (checkbox for visibility, lock button, etc)
EditorController.java	**Purpose:** Main coordinator; connects model and view. **Responsibilities:** Delegates events from the UI to the appropriate components and ensures the application state is updated accordingly.
ToolController.java	**Purpose:** Receives mouse actions from CanvasPanel. **Responsibilities:** Calls appropriate tool methods and updates the CanvasModel accordingly
FileController.java	**Purpose:** Manages file I/O **Responsibilities**: Manages importing and exporting of files. Ensures data integrity during file operations.

3. Threading

To ensure the UI is responsive, especially during intensive tasks, our design incorporates threading strategies.

Background Processing

Long running operations such as file I/O and complex image processing are executed using background threads (e.g. using Java Executor framework). This prevents the UI from freezing to

make sure the user has smooth interactions even when there's intensive processing going on in the background.

UI Responsiveness

All UI updates executed on the Event Dispatch Thread to ensure there's thread safety within swing components.

Data Consistency

When background threads modify shared data (e.g. undo) proper synchronisation mechanisms will be applied to minimise modification issues. We can use locks for this.

Design Principles

Our architecture is influenced by a set of design principles to ensure the editor is robust, maintainable and flexible. We have carefully considered each principle to address our present requirements and to facilitate future extensibility.

Separation of Concerns

We divided the application into three distinct layers using the Model-View-Controller architectural pattern. This ensures that each later handles its own specific responsibility.

Impact: Simplifies debugging and testing since functionality can be isolated. Each component can be worked on independently to allow simultaneous development – for example, UI components can be developed independently from backend logic.

Single Responsibility Principle

Each class or module is designed to have one responsibility e.g. CanvasModel only manages pixel data.

Impact: Changes in one module don't affect others (e.g. if you just wanted to update the undo mechanism).

Open/Closed Principle

The system is open for extension but is closed for modification. Using interfaces and abstract classes to ensure new features can be added without modifying existing code.

Impact: New features can be added with ease with minimal changes to existing code.

Encapsulation

Data is strictly encapsulated within the classes that manages it (e.g. ColourPalette)

Impact: There will be controlled access to data to reduce the risk of unintended side effects.

Risk Analysis

Creation Tools: These include of the colour picker, eraser, fill tool, pencil and paint brush. They are vital to the success of this project so need to be robust and reliable. The pencil, paint brush, and eraser should have varying sizes to allow for thin and thick strokes and avoid fast stokes leaving gaps on the canvas or slowing performance. The fill tool needs to be constructed well to avoid infinite loops and slow performance in the path finding algorithm that it will use. The colour picker

must handle potential errors occurring if the user selects an area outside the canvas. Whilst these tools will be easy to implement at the start of project, if they fail, the whole project fails.

Zoom: The Zoom feature will be useful for the pixel art editing feature of the project. It will not be as necessary for the Minecraft editing features as they use very low-resolution canvases. Without the zoom feature, the project could very well still be a success but, this would be a great addition. To avoid this feature having a negative impact, the zoom tool should be restricted to levels between 25% to 500%. This will improve the user experience. Additionally, the tool risks improper scaling causing distortion of the canvas, clipping with the rest of the UI, and mapping errors that may result in pixels unintentionally being drawn over. The program could also receive a performance hit if high levels of zooming out are used due to the large area that would need to be rendered.

New Colour Selector: The colour selector will be used constantly throughout the software and should allow for any colour using “hex codes” to be picked by the user. Due to this allowing for over 16 million colours, there needs to be an intuitive way for users to pick the colour they desire. Additionally, the program should save the recently used colours so the exact match can be found again easily, avoiding the need to remember hex codes or find the desired colour again. The way of assigning colours needs to be compatible with the saving feature to avoid corrupting images and files.

Undo and Redo: A vital tool for any creative software that would be used consistently but be tricky to implement. Simple solutions are possible but would restrict how many undo/redo functions can be performed and risk the user loosing previous work. Therefore, using a stack, for example, to store multiple previous states of the canvas would be far more effective albeit tricky and risky to implement. The stack of previous states could grow too large and consume memory (especially on large canvases), slowing down performance or causing a crash. Additionally, the behaviour of the stack needs to be clear to the user. For example, when does the system wipe the “redoes” it has stored. If the user doesn’t understand how it operates, they could unintentionally wipe their progress.

Flip/Rotate image: These are simple tools that would be simple to implement. Flipping an image would be useful in the Minecraft skin or block editors. Rotating an image on the other hand, certainly has more possible unintended consequences. To begin, rotating a rectangular shaped canvas would change the space on screen it requires. This could cause clipping through the UI and potentially some pixels could be lost through this. Transformations like this can also lead to misaligned pixels that do not appear where they are expected. In addition, the undo and redo features must be able to support flipping and rotating. Rotating may require storing canvas dimensions for the undo/redo tool, resulting in more memory usage.

Save/Export: Without the saving or exporting tools, the software will not be very useful. These tools will be used constantly throughout the system therefore, it is vital that they function expectedly. Otherwise, the user’s work could be lost. For saving and exporting the simple pixel art, this feature should be simple to implement robustly. The Minecraft-related projects require more caution. Minecraft textures use nets stored in the PNG format, therefore what the user draws in the software needs to correspond correctly to how the game expects it. If not done correctly, the PNGs will not work in-game and render the users work useless. Saves also need to be correctly assigned to the software’s area so they can be found easily the next time the program is opened.

Import: The import tool will allow for the users to edit existing or downloaded images. Importing images that share the system’s format will be simple. However, some constraints are required for

example, high resolution images could severely slow down the program's performance or cause crashes as could unsupported formats are should therefore not be allowed.

Minecraft Skin/Block Editor: This will utilise every other feature. It will allow, using the pixel editor, for users to create Minecraft skins that they can easily use in-game. The UI design around this feature needs to be simple and easy so that the user knows exactly where they are editing on their character. If the UI is at all confusing, the user may not create the skin they intend to. Additionally, the system needs to export the user's work in the format the game expects. Otherwise, it will be incompatible with the game and potentially lead to unintended errors such as a mismatch between what appears in our software and what appears in-game.

Feature	Likelihood of errors	Effect	Strategy
Creation Tools	Low	Serious	Avoidance
Zoom	High	Catastrophic	Avoidance
Colour Selector	Low	Insignificant	Avoidance
Undo and Redo	Moderate	Moderate	Contingency
Flip and rotate	High	Serious	Minimisation
Save and Export	Moderate	Catastrophic	Contingency
Import	Low	Insignificant	Minimisation
Minecraft Editor	High	Catastrophic	Avoidance

Acceptance tests

Test (with correlating req.)	Input	Expected Output	Unit Test Created
NF8, NF3: Compilation	Source code	1. Successful compilation using JDK 17. 2. Successful Running of it on Windows/Unix.	**N**
NF7: Window	Run program	1. Window successfully loads automatically. 2. The window dynamically updates. proportions correctly based on scaling.	**N/A**
NF2, NF5: Export	Randomised pixel data	1. Data forms a valid PNG image file. 2. PNG file matches pixel structure loaded in application memory.	**N**
F2: Undo/Redo	Multiple randomised pixel data forming states.	1. Correct state navigation forward and backward. 2. Up to and including 5 undos/redos in either direction.	**N**
F1, NF1, NF5: Image load	PNG image file;	1. PNG file matches pixel structure loaded in application memory. 2. This works when importing and when	**N**

F4.5, F6.1: Draw pixels	User draws on canvas using selection of pen sizes	1. Manipulation of pixels correlating to user's input. 2. Switching pen sizes varies the number of pixels affected when drawing, corresponding to the indicated size.	**N/A**
F4.6: Erase pixels	Users erases pre-drawn pixels	1. Selected pixels return to the default state of having no colour.	**N/A**
F6.2: Fill pixels	User selects colour; User chooses area to fill colour and clicks.	1. Colour is propagated using simple path finding algorithm of those pixels of the same colour to selected and on same x and neighbouring y, or same y and neighbouring x. 2. No colours are propagated diagonally	**N**
F6.3: Change pen colour	User creates custom colour; User chooses colour; User draws on canvas.	1. The selection of pixels by the user turns the correct colour. 2. Custom colour is saved and can also be selected and used as described in (1).	**N/A**
F3, NF4: Screen reader	Application loaded on both Windows/Ubuntu; Windows native screen reader and Orca screen reader are turned on for their respective OS.	1. All text can be accessed by screen reader. 2. Text is read out correctly. 3. Non-custom colours and icons are labelled when hovered over. 4. Text size is adjustable.	**N/A**
F5, F6.4: Colour scheme	User defines RGB for colour; User selects colour from colour palette.	1. Correct colour is represented based on RGB input.	**N/A**
F6.3: Colour picker	User uses colour picker to select colour.	1. Selected colour becomes their primary colour.	**N/A**
F6.5: Blur	User varies blur % of image.	1. The higher the percent the greater the loss of detail in the image. 2. The processing of blurriness is correctly transformed into exported image.	**N**
F4.1: Layer selector	User creates a new layer; User draws on new layer.	1. New layer is created and able to be selected. 2. Drawing is only applied to this layer.	**N**
F4.2: Flip	User selects flip horizontally; User selects flip vertically.	1. Horizontal: pixel matrix is correctly reflected over y-axis. 2. Vertical: pixel matrix is correctly reflected over x-axis.	**N**
F4.2: Rotate	User selects rotate.	1. Image is rotated 90 degrees to the right.	**N**
F4.3: Cropping	User selects cropping tool; User adjusts crop	1. The crop is successfully applied removing the	**N**

	square to fit chosen size; User applies change.	unselected areas from the image	
F8: Grid	User toggles on/off grid overlay.	1. Grid scaled to proportional size of image is overlayed when turned on. 2. Grid is removed when turned off. 3. Not propagated to export/save.	**N/A**
F9: Zoom	User varies zoom %.	1. Correct size of image is shown based on zoom level i.e. if 500% it should show 1/5 of the image.	**N/A**
F4.4: Pixel selection	User selects pixel selector tool; User drags mouse over pixels; User applies action to pixels.	1. Correct pixels are highlighted. 2. Correct action is applied to the selected pixels.	**N/A**
F10.1: Minecraft skin	User designs Minecraft skin; User exports to PNG.	1. The file successfully exports to valid png. 2. The image is able to be loaded into Minecraft successfully with accurate skin design.	**N**
F10.2: Minecraft block	User designs Minecraft block; The user exports the block to PNG.	1. The file successfully exports to valid png. 2. The image is able to be loaded in as a texture pack with accurate block design.	**N/A**
NF6: Keyboard shortcuts	The user uses CTRL+S; The user uses CTRL+C/CTRL+V on a selection of pixels.	1. The system saves the file 2. The pixels are copied and pasted to another part of the canvas	**N/A**

Task List

Week 12	We introduced ourselves and created a plan for D0. These features were dark mode and JSON exporting.
Week 13	We discussed our ideas and settled on creating a pixel editor for Minecraft. We were able to choose this due to our joint interests and skills. We then assigned sections and responsibilities within the design report.
Week 14	We did a review of the progress made, this helped us to discuss difficulties we were facing and make sure our overall plan was meeting requirements and accomplishable.
Week 15	We got feedback from tutors and were able to realign to make sure we were meeting all the

	goals. After submission we plan to make the basic window and canvas functions.
Week 16	Having created our design tasks, we will assign implementation tasks and work on building the basic structure/functionality in line with our architecture design. The tasks will include making the basic tools for the application such as the brush and eraser.
Week 17	Having completed our basic functionality, it will be time to implement GitLab tests to run to validate our program meets the acceptance tests detailed above.
Week 18	By this point we should have an MVP, which means it's time to revise our previous work to make sure it remains well structured and we are meeting requirements. Any blockers will be tackled during our weekly team meet-up.
Week 19	Since out work is refined, we can now work on some more complicated functionality and some especially interesting ones which involve matrix calculations such as rotate or the fill tool.
Week 20	By this point we hope to have met our 3rd milestone meaning we will be focussing less on building the architecture and more on expanding features. Another thing we will focus on is updating GitLab tests considering the new features we have implemented.
Week 21	The new GitLab tests may cause some issues to arise, therefore we will work as a team to remove the mitigations. Lastly It will be time to work on unique features that allows users to create Minecraft skins and blocks that can be shared with other users.

Tasks Checklist:

Task	**Deadline (recalculate based on weekends)**	**Completed (Y/N)**
GitHub test to verify compilation	15/2/25	**N**
Create window	17/2/25	N
Create canvas	18/2/25	N
Create toolbar and colour bar	18/2/25	N
Create brush and eraser	21/2/25	N
Create undo/redo	24/2/25	N
Save image	24/2/25	N
Open image	24/2/25	N
Create basic tests (based on acceptance tests for previous features).	3/3/25	N
Allow changing colours	4/3/25	N

MVP refining previous features	9/3/25	N
Flip/rotate image	11/3/25	N
Varying brush sizes	11/3/25	N
Fill tool	12/3/25	N
Multiple Layers	17/3/25	N
Crop	19/3/25	N
Zoom	19/3/25	N
Select Pixels	20/3/25	N
Apply tools to selected pixels	21/3/25	N
Blur	21/3/25	N
Update GitLab tests	24/3/25	N
Resolve issues from tests	28/3/25	N
Minecraft Canvas'	29/3/25	N
Export Minecraft blocks	3/4/25	N

Project Plan

ID	Milestone	Deliverables	Deadline (End of Week
	Milestone #1		**Week 17**
M1	Create window	Java class to facilitate the initialisation and management of the Swing window.	**15**
M2	Create canvas	Java class to facilitate the initialisation and render of the editable frame within the window.	**15**
M3	Create toolbar and colour bar	Java class that renders a drawable canvas area within the application window.	**16**
M4	Create brush and eraser	Java class that allows brush and eraser tools.	**16**
M5	Create undo/redo	Java class that provides an undo/redo stack for actions performed on the canvas.	**16**
M6	Save image	Java class that allows exporting the current canvas to a PNG image file.	**16**
M7	Open image	Java class that allows importing an existing image file on to the canvas.	**16**
M8	Create basic tests (based on acceptance tests for previous features). Should be updated if a new feature affects them and added to GitLab on pull requests.	Create new and amended Java test class files that validate the previously implemented features.	**17**
	Milestone #2		**Week 19**
M9	Allow changing colours	Java class that adds the ability to change the current drawing colour using a colour palette.	**18**

M10	MVP refining previous features	Amended Java class files that add the ability to change the current drawing colour dynamically in the colour palette.	**18**
M11	Flip/rotate image	Java class that allows flip and rotate transformations for images on the canvas.	**19**
M12	Varying brush sizes	Expanded Java Class that allows brush tool to support multiple brush sizes.	**19**
M13	Fill tool	Java class files that implement a fill tool, allowing users to fill contiguous areas of the canvas with a chosen colour.	**19**
M14	Multiple Layers	Amended Java class files that adds layer functionality within the canvas.	**19**
	Milestone #3		**Week 21**
M15	Crop	Java class that implements a cropping function, allowing users to select and trim parts of the image.	**20**
M16	Zoom	Java class that enables zoom in/out capabilities to the canvas view.	**20**
M17	Select Pixels	Java class that enables pixel-level selection, enabling the highlighting of specific regions on the canvas.	**20**
M18	Apply tools to selected pixels	Java class that adapts existing tools (brush, eraser, fill, etc.) to operate only on a selected pixel region.	**20**
M19	Blur	Java class implementing a blur effect that can be applied to selected region or entire layer.	**20**
M20	Update GitHub tests	Utilising JUnit we can run tests every time new code is pushed to the repository…	**20**
M21	Resolve issues from tests	Refined/Amended Java class files that (bug fixes, improvements) address failures or issues identified during our tests.	**21**
M22	Minecraft Canvas	Java class files that introduce a Minecraft-styled canvas, including a pixel grid to create textures for the game	**21**
M23	Export Minecraft blocks	Java class that export's a canvas into a valid Minecraft texture pack format.	**21**

Activity Network with Critical Path

This defines the dependencies of each task on one another. The critical path is the absolute set of features, in which if a task is delayed, it affects the rest of the tasks around it. It is highlighted in yellow in the appendix (2.B).

Gantt Chart

Can be seen below (Appendix 2.A).

Costings (See **Appendix 3**)

	Cost per hour (pp)	Estimated Hours (all)	Estimated Sub-total
Planning and Design phase (w13-15)		52	£1,126.95
Team on-boarding (induction, technical setup)	£20.55	9	£184.95
Initial topic selection	£20.55	6	£123.30
Feature selection and principles	£20.55	6	£123.30
Requirements analysis	£20.55	3	£61.65
UI design	£40.00	3	£120.00
Architectural design	£20.55	3	£61.65
Risk analysis	£20.55	2	£41.10
Acceptance tests	£20.55	2	£41.10
Schedule planning (tasks, activity & Gantt)	£20.55	6	£123.30
Design report writing and editing	£20.55	12	£246.60
Implementation phase (w15-21)		386	£16,333.00
Milestone 1 development (w15-17)		132	£5,496.00
Design (class design)	£50.00	12	£600.00
Development (coding)	£50.00	48	£2,400.00
Documentation	£50.00	16	£800.00
Testing	£15.50	32	£496.00
Coordination (additional meetings, meetings)***3A**	£50.00	24	£1,200.00
Milestone 2 development (w17-19)		108	£4,572.00
Design	£50.00	12	£600.00
Development	£50.00	36	£1,800.00
Documentation	£50.00	12	£600.00
Testing	£15.50	24	£372.00
Coordination (additional meetings, meetings)	£50.00	24	£1,200.00
Milestone 3 development (w19-21)		146	£6,265.00
Design	£50.00	12	£600.00
Development	£50.00	60	£3,000.00
Documentation	£50.00	20	£1,000.00
Testing	£15.50	30	£465.00
Coordination (additional meetings, meetings)	£50.00	24	£1,200.00
User Evaluation phase (w21-22)		158	£6,761.50
Evaluation design	£50.00	18	£900.00
Ethics approval process	£50.00	5	£250.00
Participant recruitment	£50.00	9	£450.00
Running evaluation	£15.50	15	£232.50
Analysis	£50.00	12	£600.00
Resulting changes		99	£4,329.00

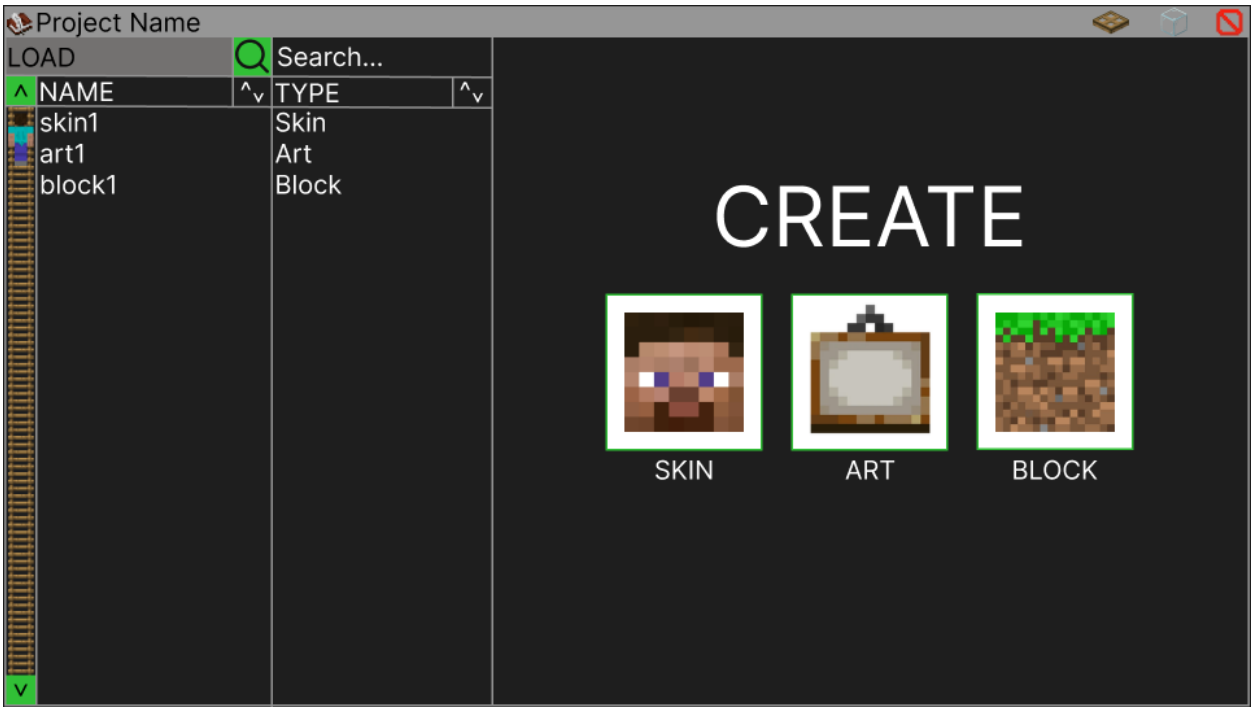
Design	£50.00	12	£600.00
Development	£50.00	36	£1,800.00
Documentation	£50.00	9	£450.00
Testing	£15.50	18	£279.00
Coordination (additional meetings, meetings)	£50.00	24	£1,200.00

Non-staff costings	**P&P *3B**	**M1**	**M2**	**M3**	**User Eval*3 C**	**Sub-total by cost**
Developer equipment (computers)	£5,000.00	£0.00	£0.00	£0.00	£0.00	£5,000.00
Hosting costs	£0.00	£0.00	£0.00	£0.00	£0.00	£0.00
Software subscriptions (GitLab)	£348.00	£0.00	£0.00	£0.00	£0.00	£348.00
User eval participant costs	£0.00	£0.00	£0.00	£0.00	£125.00	£125.00
Sub-total by phase	£5,348.00	£0.00	£0.00	£0.00	£125.00	

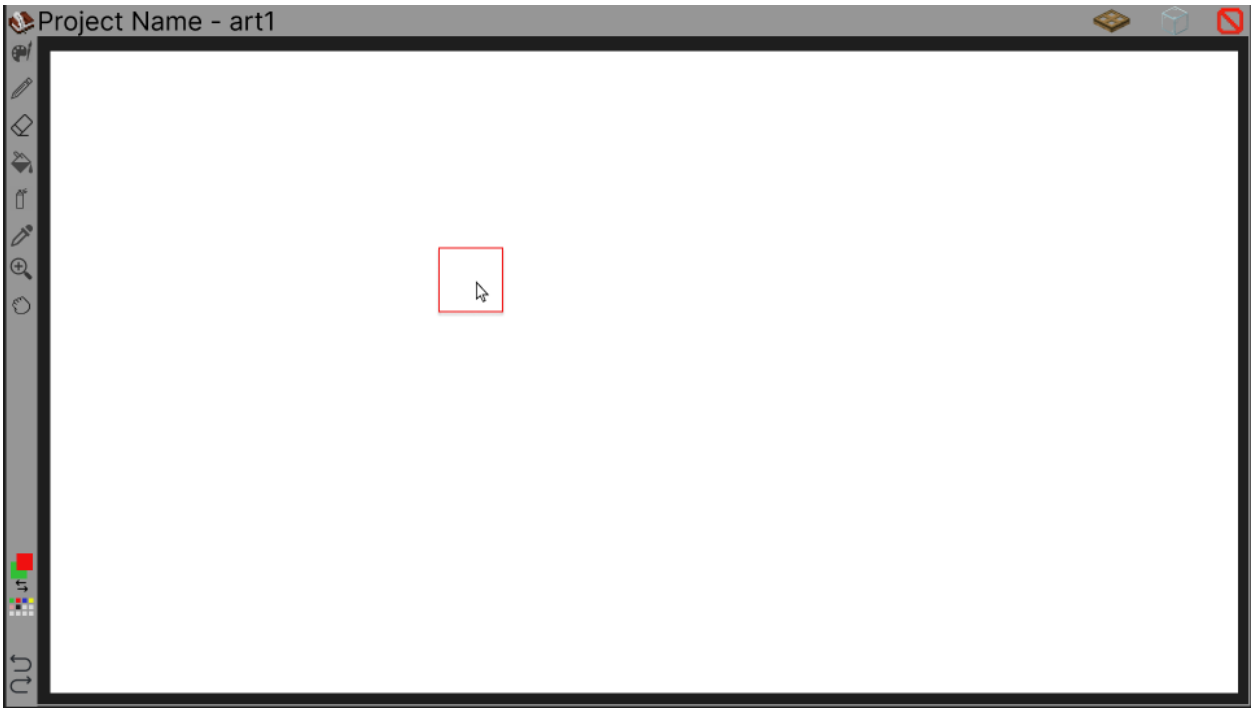
Total person-hours effort	596
Total staff costs:	£24,221.45
Total non-staff costs:	£5,473.00
Total project cost estimate:	**£29,694.45**

Appendix

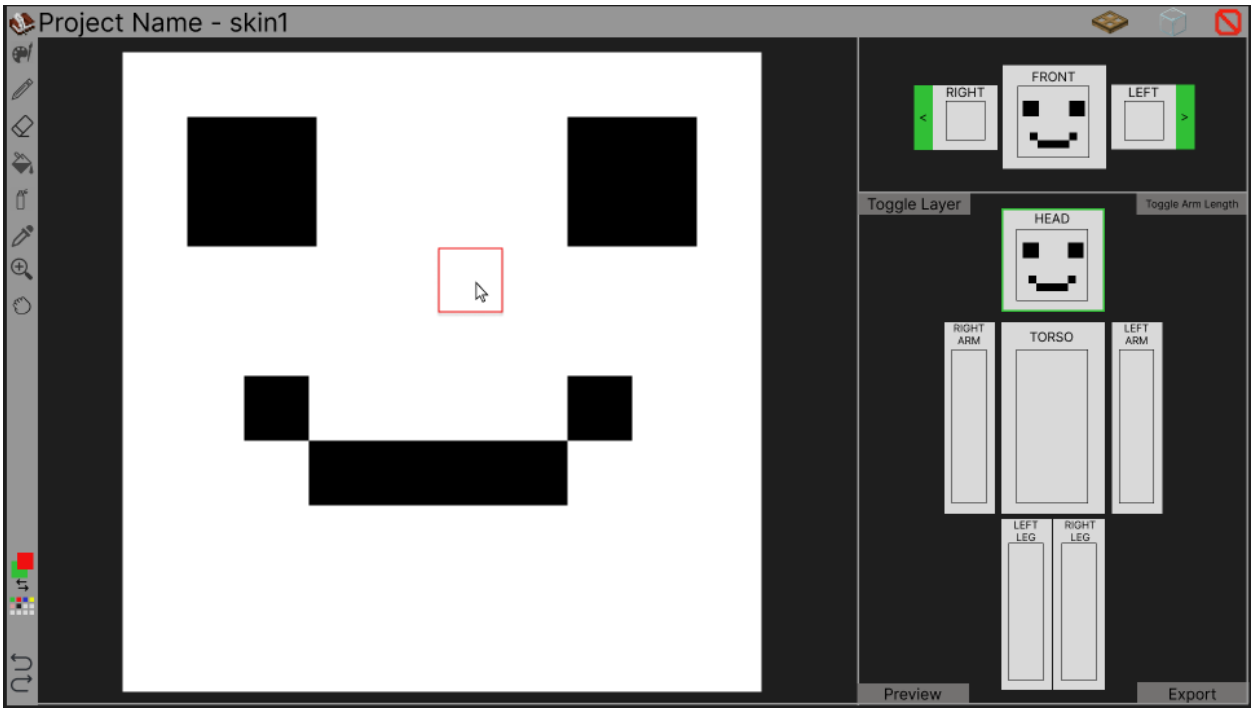
1.A)


Project Name
LOAD
Search...
NAME
TYPE
skin1
Skin
art1
Art
block1
Block
CREATE
SKIN
ART
BLOCK

1.B)


Project Name - art1

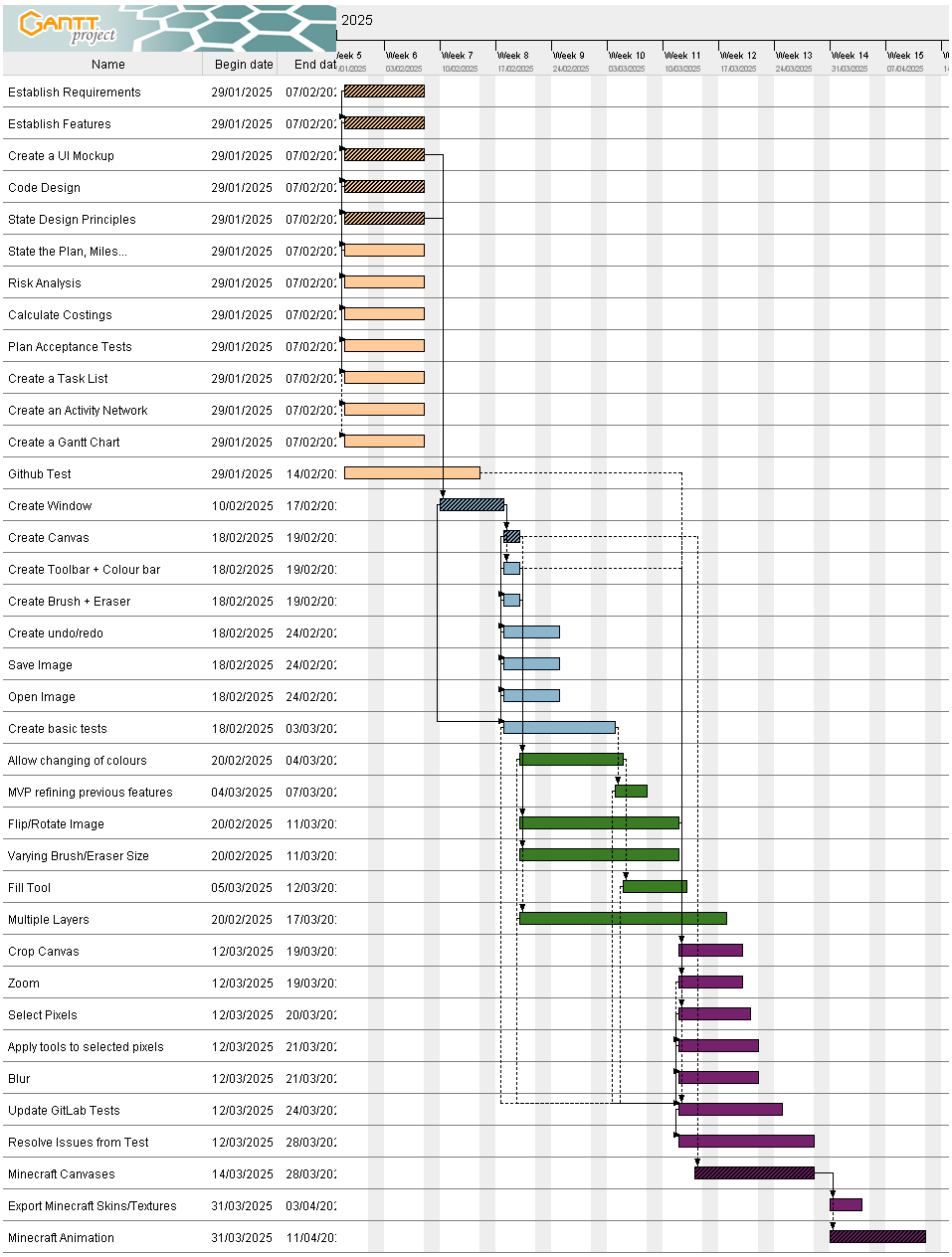
1.C)


Project Name - skin1
RIGHT
FRONT
LEFT
Toggle Layer
Toggle Arm Length
HEAD
RIGHT ARM
TORSO
LEFT ARM
LEFT LEG
RIGHT LEG
Preview
Export

1.D)

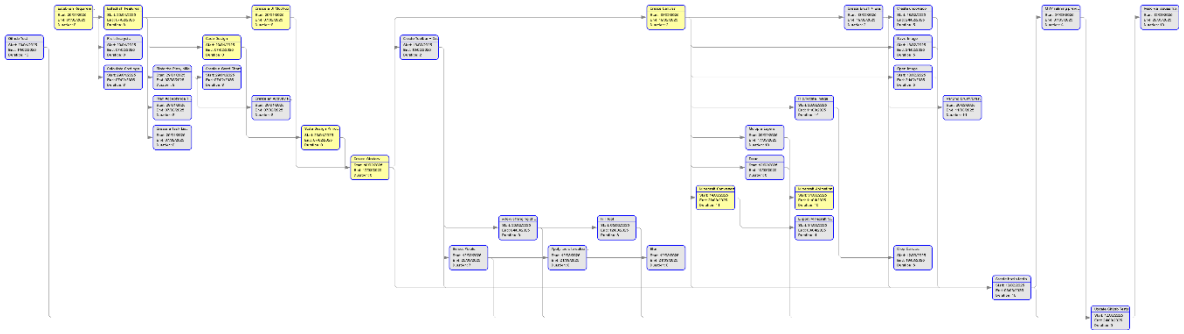

Project Name - block1
TOP
FRONT
LEFT
BOTTOM
BACK
RIGHT
Preview
Export

2.A)



Name	Begin date	End dat
Establish Requirements	29/01/2025	07/02/20
Establish Features	29/01/2025	07/02/20
Create a UI Mockup	29/01/2025	07/02/20
Code Design	29/01/2025	07/02/20
State Design Principles	29/01/2025	07/02/20
State the Plan, Miles...	29/01/2025	07/02/20
Risk Analysis	29/01/2025	07/02/20
Calculate Costings	29/01/2025	07/02/20
Plan Acceptance Tests	29/01/2025	07/02/20
Create a Task List	29/01/2025	07/02/20
Create an Activity Network	29/01/2025	07/02/20
Create a Gantt Chart	29/01/2025	07/02/20
Github Test	29/01/2025	14/02/20
Create Window	10/02/2025	17/02/20
Create Canvas	18/02/2025	19/02/20
Create Toolbar + Colour bar	18/02/2025	19/02/20
Create Brush + Eraser	18/02/2025	19/02/20
Create undo/redo	18/02/2025	24/02/20
Save Image	18/02/2025	24/02/20
Open Image	18/02/2025	24/02/20
Create basic tests	18/02/2025	03/03/20
Allow changing of colours	20/02/2025	04/03/20
MVP refining previous features	04/03/2025	07/03/20
Flip/Rotate Image	20/02/2025	11/03/20
Varying Brush/Eraser Size	20/02/2025	11/03/20
Fill Tool	05/03/2025	12/03/20
Multiple Layers	20/02/2025	17/03/20
Crop Canvas	12/03/2025	19/03/20
Zoom	12/03/2025	19/03/20
Select Pixels	12/03/2025	20/03/20
Apply tools to selected pixels	12/03/2025	21/03/20
Blur	12/03/2025	21/03/20
Update GitLab Tests	12/03/2025	24/03/20
Resolve Issues from Test	12/03/2025	28/03/20
Minecraft Canvases	14/03/2025	28/03/20
Export Minecraft Skins/Textures	31/03/2025	03/04/20
Minecraft Animation	31/03/2025	11/04/20

2.B)



3) All costs per hour are based on the hourly rates suggested from payscale.com. All tasks except UI design in the planning and design phase are designated to ‘Design analyst’ roles and are paid £20.55/hour. UI design is designated to ‘UX designer’ roles, paid £40.00/hour.

In the implementation stage, all ‘Software developers’ are allowed £50.00/hour. Testing tasks are allocated to ‘Test/QA Engineers’, paid £15.50/hour.

All hours are cumulative in the case where multiple people work on the same task together.

- **A.** We expect all 6 members to meet an hour a week, so 24 hours are allocated to coordination every (2-week) milestone.
- **B.** We allow £5000 to purchase computers and equipment for the team to allow roughly £833 for each person. We also allocate £348 for two months’ GitLab subscription to facilitate version control.
- **C.** We set aside £125 as incentive for participants to take part in testing. (As of now, expected £5 for 25 participants)