# Sketch Based Animation System

**ACM Reference Format:**
. 2024. Sketch Based Animation System. 1, 1 (May 2024), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## INTRODUCTION

Our team has opted for a sketch based animation system, which is a UI based tool that lets the user draw and animate their drawings on the canvas.
In this Project, Our team has implemented a UI based sketching system which helps in sketching dynamic and interactive illustrations. Artists can sketch animated drawings and textures to convey the living phenomena or specify working of machines or just have fun.

## MILESTONES ACHIEVED

- User-Interface: Main Canvas, Global Toolbar.
- Enabling Users to draw on Canvas.
- Select object and add Basic Transformation to Object.
- Emitting Texturing.
- Oscillating Texturing.
- Tool for creating animated Objects.
- Graph Mode.
- Cause and Effect Animation.
- Presentation Mode.

## GITHUB PROJECT CODE LINK

Sketch Based Animation GitHub URL - https://github.com/CSE-333-Computer-Graphics-2020/SketchBasedAnimation

## LIBRARIES USED

- **Dear ImGui** : For Creating Graphical User Interface
- **GLM** : For Matrix Operations
- **STB** : For Image Texturing

## ALGORITHM AND IMPLEMENTATION DETAILS

(1) **Drawing Canvas**
We have implemented drawing canvas with the use of mouse position and mouse button state mode. We have captured all the X and Y coordinates that the mouse cursor pass through while the left mouse button is

Author's address:

XXXX-XXXX/2024/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

pressed and stored it in a dynamic array of 2D-vectors. When the user presses down the left mouse button, a flag is set and new object is created to store all the details of this object and perform basic functionalities. On releasing the mouse button we reset the flag so as to let the program know to stop adding points to the object shape. We have used Poly-Lines to create each object.

(2) **Selection of Strokes**
We have created a selection buffer by using a 2D-Array which stores index of all the objects drawn on the screen at their respective $(X, Y)$ coordinates of their poly-line points, with ±5 pixels for easier selection. On-click the code accesses the selection buffer and fetches the index of the object under the cursor. Now the user is ready to apply transformations to the object.

(3) **Post selection transformations**
In our implementation, we have considered list of points as a object. We have created a Class for the object and defined member functions translate, rotate and scale for it. Object can perform these transformations upon users click. For transformations requiring a pivot(anchor), when the user clicks $Transform$ button, the transformation message changes to "Please select anchor point" after which the next click that the user makes on the *Canvas* will be saved as the anchor point, and then the user can perform desired transformations.

(4) **Switching between draw and selection modes**
For switching between "Drawing" and "Selection" modes we have created 2 buttons which changes the values of a variable so as to specify the current mode.

(5) **Basic Transformation of Objects**
We have implemented a class named "PlObject" to hold all the data of the object, including the array of points that it passes through and the basic transformations that can be applied to this object. We invoke these functions using the object of the shape. The arguments that are passed to these transformation functions are set and modified by the slider components in the toolbar. On clicking the "Transform" button, the set transformations take effect.

(6) **Kinetic texturing**
We have implemented a class to hold the basic data structure and common functionalities required by both "Emitting Texture" and "Oscillating Texture". The emitting and oscillating texture classes will inherit these properties and build on them to give desired functionality.
These classes will use the texture mapping concept along with animation on shapes to give their desired functionality.

(7) **Emitting Texturing**
We have implemented Emitting Textures by using set of points, that represent the emitting source object that we stored while drawing. For every frame we are creating an emitting object having mid-point on a random point from the set of source object points. We are also transforming the emitted objects for every frame by some pixels in the next frame, we are also checking if any of these objects have gone out of canvas or not after every transformation. If it's out of canvas then we destroy that object.

(8) **Oscillating Texturing**
In Oscillating texturing, the selected object that we wish to oscillate between current position of the object

and a chosen point anywhere on canvas, we divide its path into parts and translate the object to each part in every iteration to make it oscillate back and forth between these two positions.

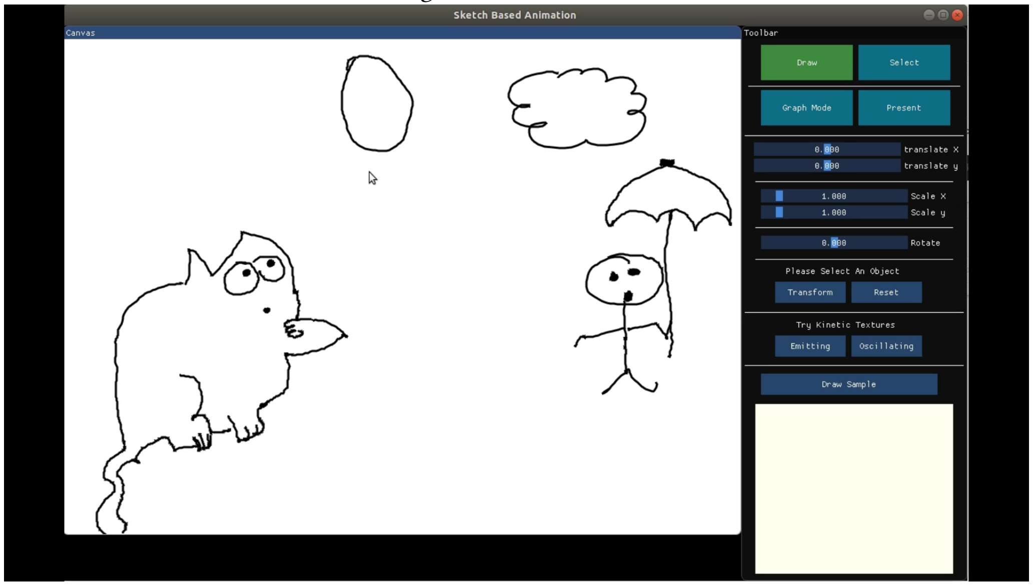
(9) **Graphs Mode Feature**
In the graph mode, we have implemented a rectangular component in ImGui and have mapped a texture of Grid Lines over it. so as to give better UX experience while setting initial and final positions of their sketched components. We have provided a timeline slider for the user to set initial and final position of the sketches. The user can select objects and transform their positions in the graph mode as well excluding the rotation and scaling operations. Based on the position of the timeline slider, we save the position of each object in the sketch. Both the initial and final state of the sketch will be used in cause and effect animation in presentation mode.

(10) **Cause and Affect Animation**
When we interact with one object then other objects which has been set to respond to change in the selected object, also change their state accordingly. We have implemented a Class named Timeline which store the initial and final position of the sketch through graph mode. When a object is selected and dragged towards its final state, the complete state of the sketch moves towards the final state. This enabled us to interpolate animation between initial and final state of the sketch.

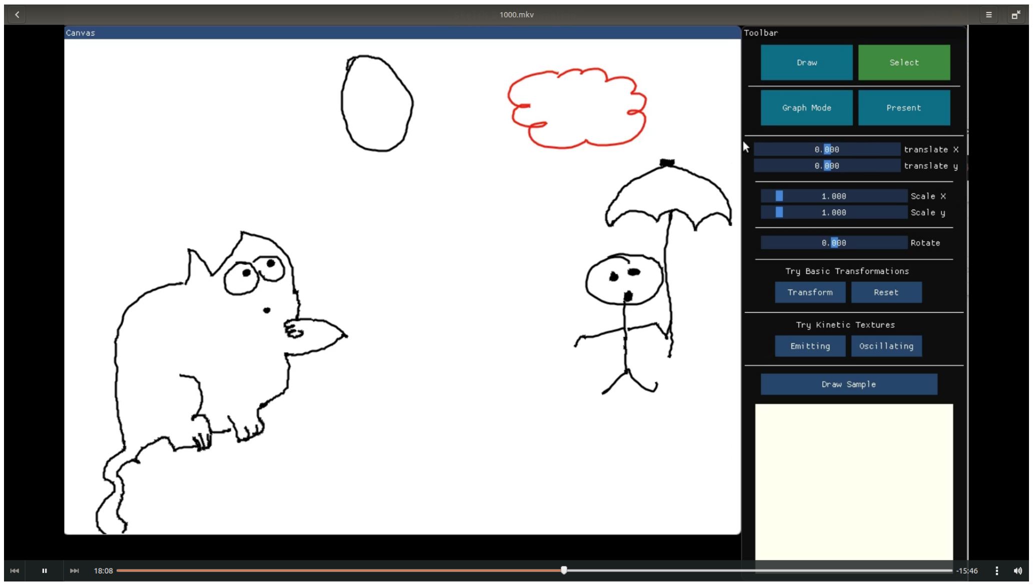
(11) **Presentation Mode**
In presentation mode, the user can interact with an object to witness the interactive animation that he or she has set to make the sketch come to life. After selecting an object, when the user starts to drag the object in any direction we store the click coordinates and the current coordinate to form a temporary vector, we also created a set of vectors named "directions" which starts at the initial position of each object and ends at the final position of the respective object. We calculated the dot product of the temporary vector and direction vector of the selected object to find the projection of temporary vector onto direction vector which gives us an idea of how much the objects needs to be transformed towards the final state. This also prevents the object moving from initial to final state, when the user has dragged the object in opposite direction.
In this mode, we have also implemented a function which calculates the fraction state by dividing the above calculated dot product by the magnitude of direction vector of selected object. We also used this fraction state to calculate the current position of each object and display them.

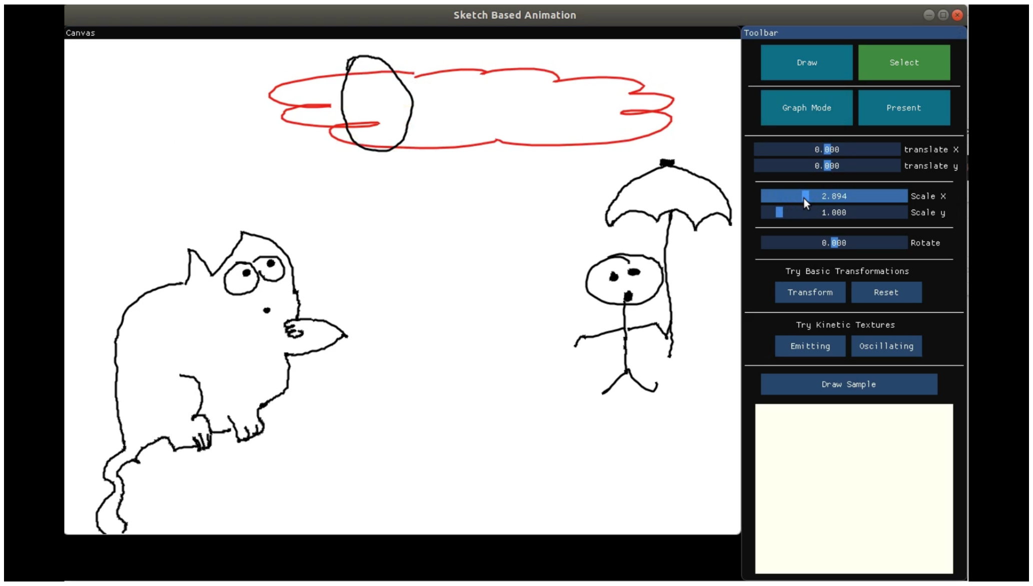
## SCREENSHOTS

**STEP 1** : Draw Sketch on Canvas using Draw mode



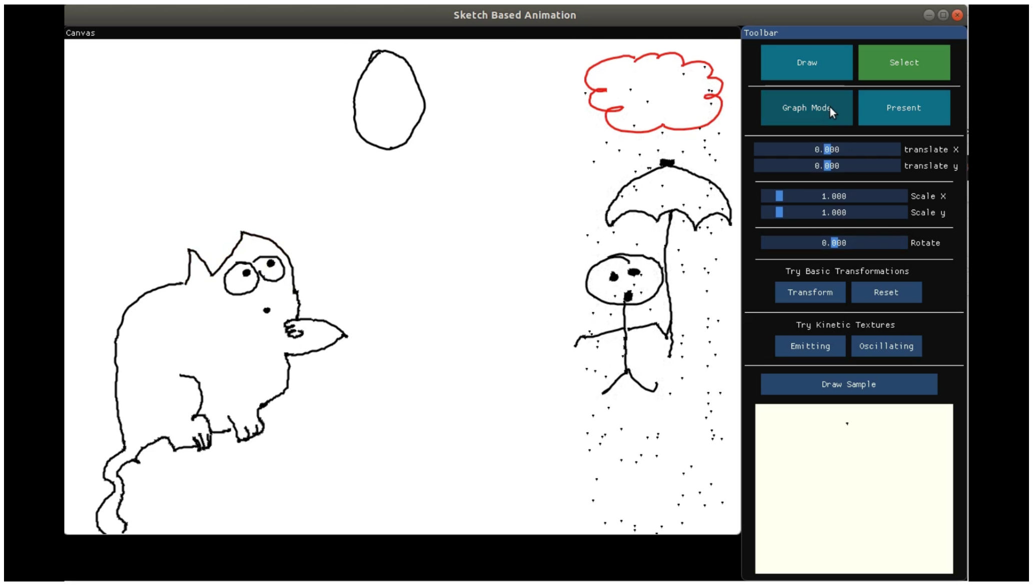
**STEP 2** : Select the desired strokes USING SELECT MODE and perform transformations like TRANSLATION, ROTATION (about arbitrary point), SCALING (can scale about arbitrary point) to adjust the sketch elements accordingly.





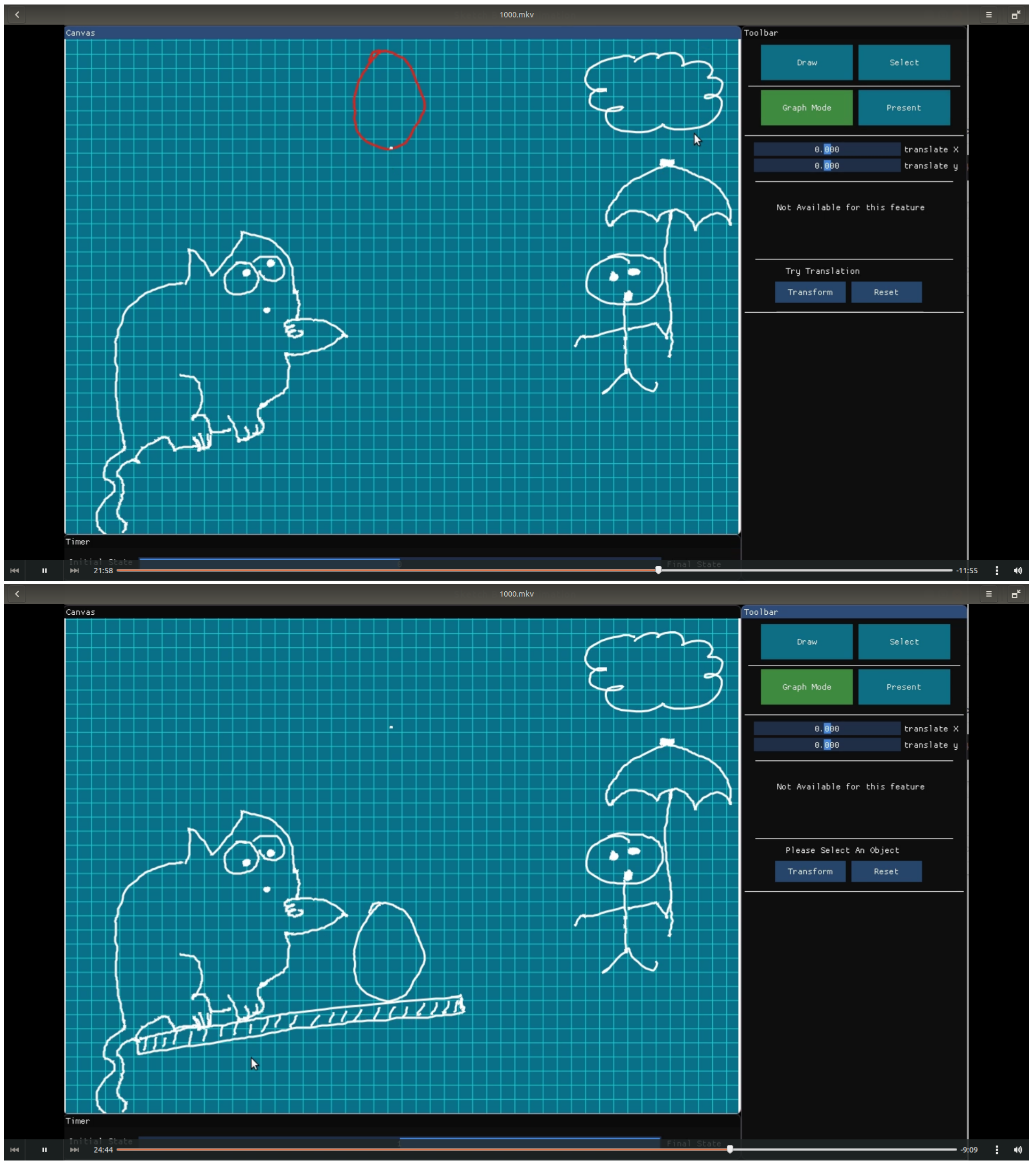
0.000 translate X
0.000 translate y
2.894 Scale X
1.000 Scale y
0.000 Rotate

**STEP 3** : In this sketch, we select the cloud and apply EMITTING TEXTURE effect which gives you rain effect.

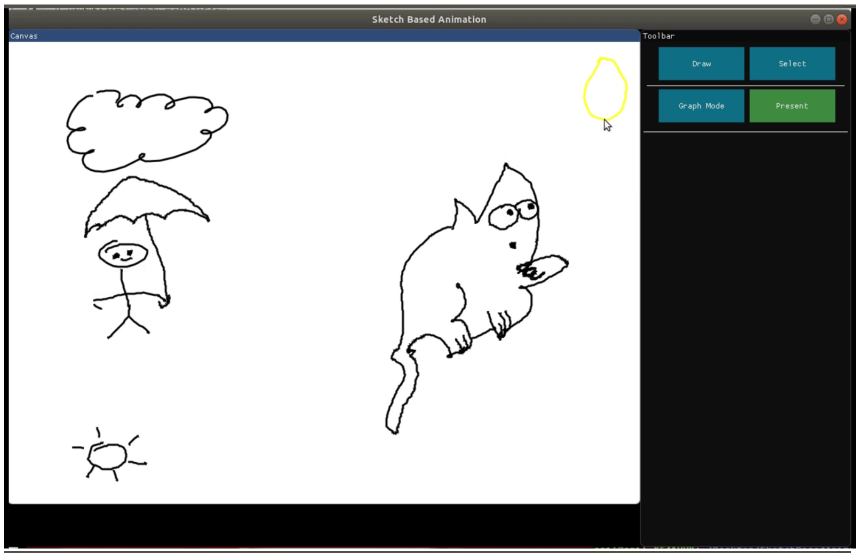


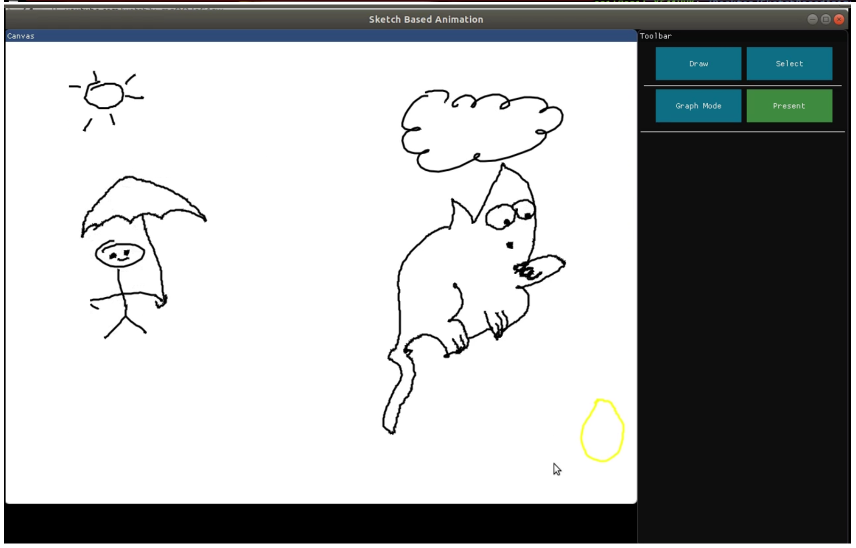
0.000 translate X
0.000 translate y
1.000 Scale X
1.000 Scale y
0.000 Rotate

**STEP 4** : Now, You Select GRAPH MODE, it will let you translate the elements in sketch and SAVE in two different timelines ( TIMELINE BAR at the Bottom)



**STEP 5 (PRESENT SKETCH)** : After you have set the sketch in GRAPH MODE, you can see things moving in PRESENT MODE. As you move one element of the sketch , the other connected elements will move too.





## CONRIBUTIONS

- **Suraj Rathore** : Main Canvas, Toolbar, Enabling User to draw on Canvas, Object Selection, Emitting Texture, Cause and Effect Animation, Report(Implementation Details)
- **Nishkarsh Saxena** : Emitting Texture, Tools for Applying animation to Object, Graph Mode, Github Repository, Report(Algorithm Implementation)
- **Anil Kumar** : Trasformations - Translation, Rotation, Scaling, Oscillating Texture, Report(Screenshots, Implementation Details)

## CITATIONS AND BIBLIOGRAPHIES

(1) Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. (2014) Kitty: sketching dynamic and interactive illustrations. ACM NY, USA, 395–405.
(2) Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S.  Fitzmaurice, G. (2014) Draco: Bringing Life to Illustrations with Kinetic Textures. ACM CHI, 351-360.
(3) Lecture slides, Labs code, Assignment codes,Lecture videos