Python Desktop & Mobile Development

Table of Contents

[Objectives: 2](#_Toc154869700)

[There will be 3 steps: i.e., V1, V2 & V3 2](#_Toc154869701)

[Game will be Modified to Chandra Yan: 2](#_Toc154869702)

[V1 Layouts: 2](#_Toc154869703)

[In V1: 2](#_Toc154869704)

[In V1 Continued: 4](#_Toc154869705)

[In V1 Horizontal Lines Perspective Continued: 5](#_Toc154869706)

[V2 Widgets: 8](#_Toc154869707)

[In V2 Continue: Structure the Code: 8](#_Toc154869708)

[In V2 Continue: Tiles and Line Coordinates: 9](#_Toc154869709)

[V3 Canvas: 14](#_Toc154869710)

[In V3 canvas -> Game Over state 14](#_Toc154869711)

[Tools and Sources Used: 14](#_Toc154869712)

# Objectives:

1. Practice on apps development with Kivy (canvas)

2. Learn to structure and organise the code.

3. Improve the debug skills (errors, wrong behaviours)

4. Practice on algorithms development (land generation, perspective display)

5. Create and implement the game elements (land, ship.)

6. Code the game logic (User actions, game over, score, sounds…)

# There will be 3 steps: i.e., V1, V2 & V3

1. In V1 layouts, Displaying the grid in 2 dimensions, & transforming the Grid in Perspective and adding the Movements. Movement means to go forward and go left and right.

2. In V2 Widgets, we will generate the land (track the spaceship must follow), display the ship, Manage the collision, we will observe the ship if it’s out of then track or in the track in that case, we can display Game Over.

3. In V3 Canvas, we will display the Menu that triggers the different sounds, display the background and Score.

# Game will be Modified to Chandra Yan:

1. Spaceship as a Chandra Yan.

2. Mission: Shiva Shakti

3. Background: To Moon

4. Sound: Shiva Tandav or Mritunjiya Mantra, Om, Namah: Shivayey:

5. Avoiding the classics.

# V1 Layouts:

- We are going to create a project to define the coordinates of prospective point.

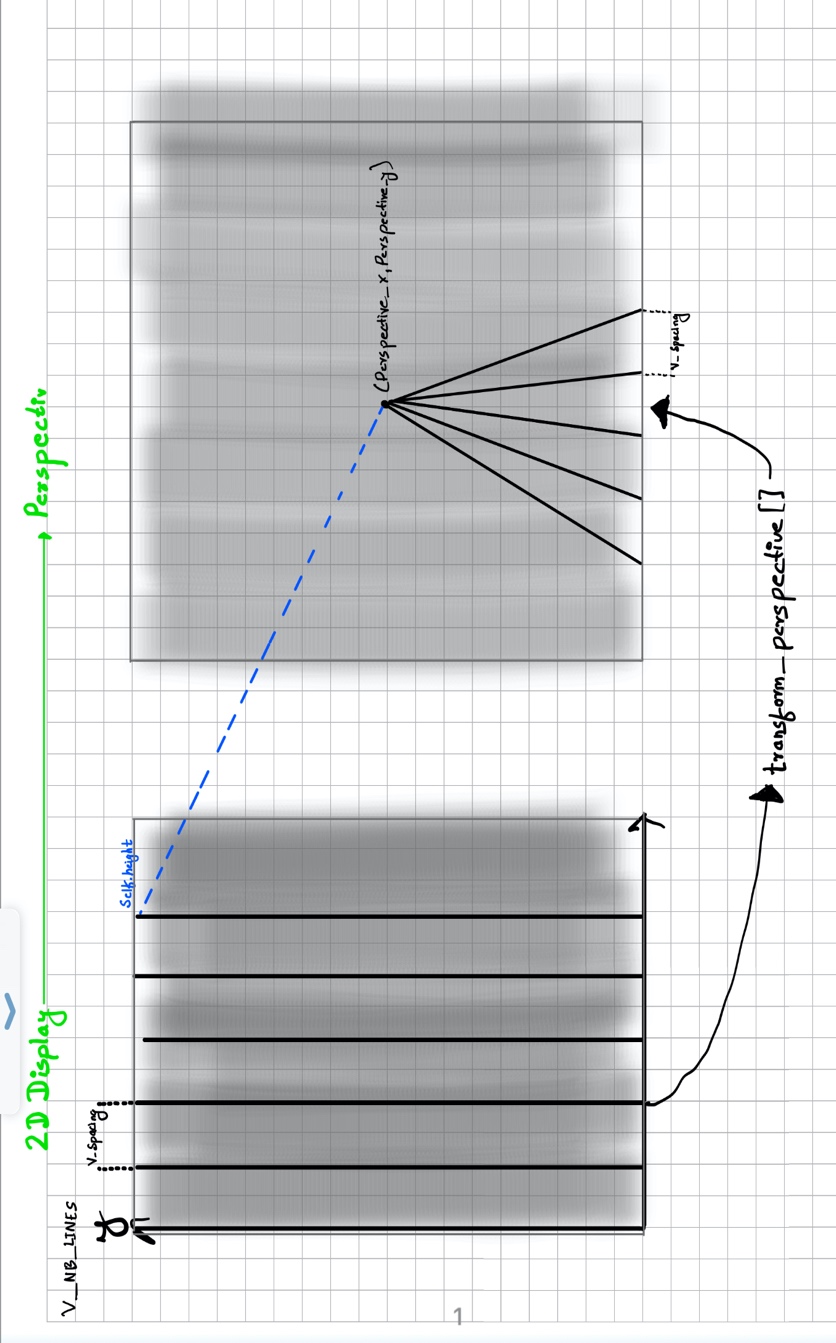
- We are going to generate the grid with vertical and horizontal lines.

- We are also going to create transformation of coordinates that we can go from 2 degree to the perspective mode.

- We are also going to implement forward, left, and right actions.

## In V1:

* We were able to display the vertical lines in 2-dimensional space.
* We have five Vertical Lines which is V\_NB\_LINES.
* We use V\_SPACING (Vertical Line Spacing) to define the equal spacing between the different lines.
* Vertical Lines start from y of 0 to height of the screen which is self.height.
* Next thing to do is to transform the 2D display to perspective display.
* We already have perspective\_x and perspective\_y.



* We are going to put the same value for the Line Spacing.
* The closer we go to the vertical points; the horizontal axis is going to be smaller.
* Now we can transform it into perspective mode by implementing transform\_perspective[] function.
* We are using Mathematical computation which is linear proportions to go from 2D to Perspective display.
* It will take (x,y) 2D coordinate as a input and return the (x,y) TR transform coordinate.
* On the Lines, point no 1 and point no 2; point no 1 will be displayed at the same position whereas point no 2 will be displayed at the same position as (perspective\_x, perspective\_y)

## In V1 Continued:

A graph of a graph on a graph paper

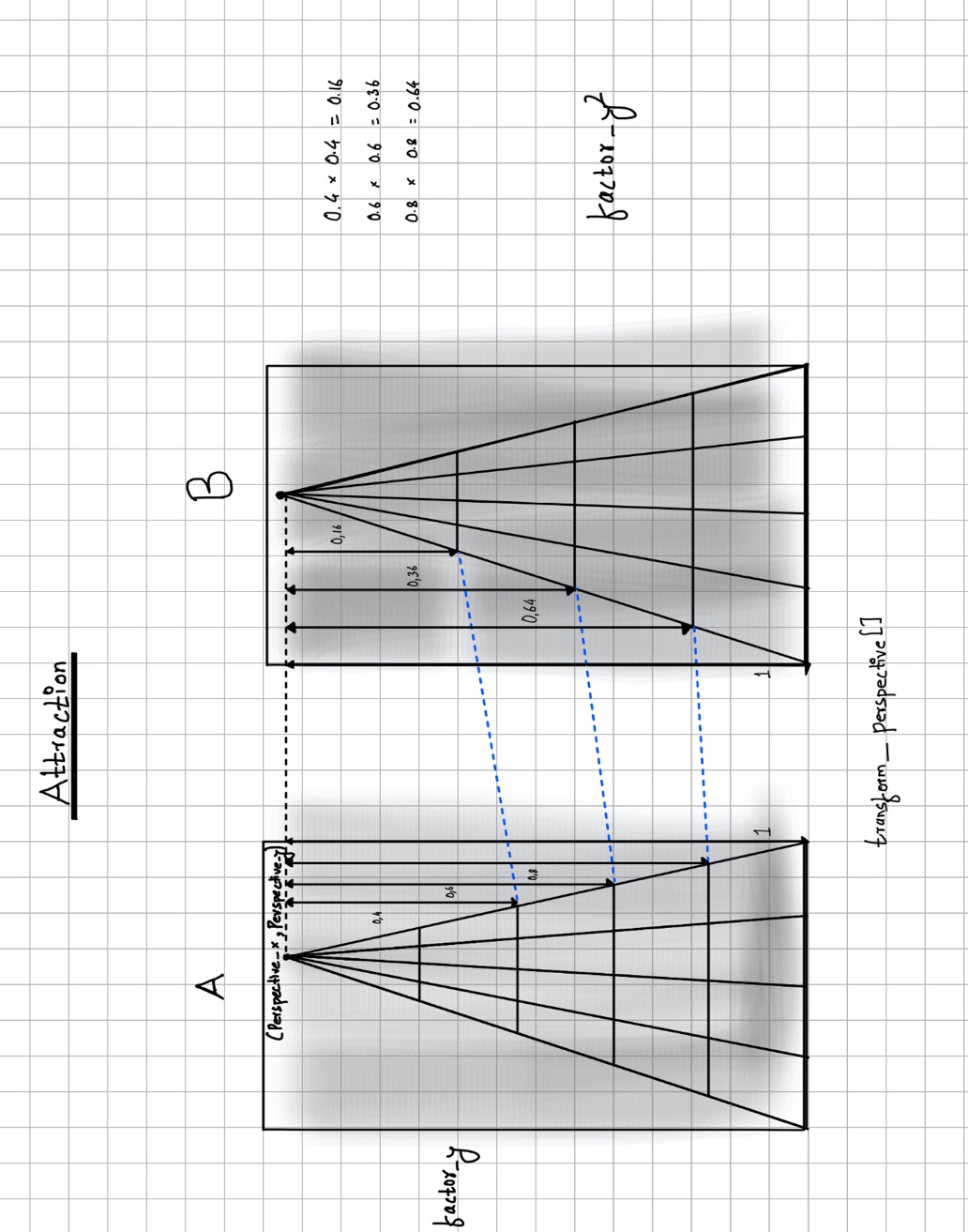
Description automatically generated

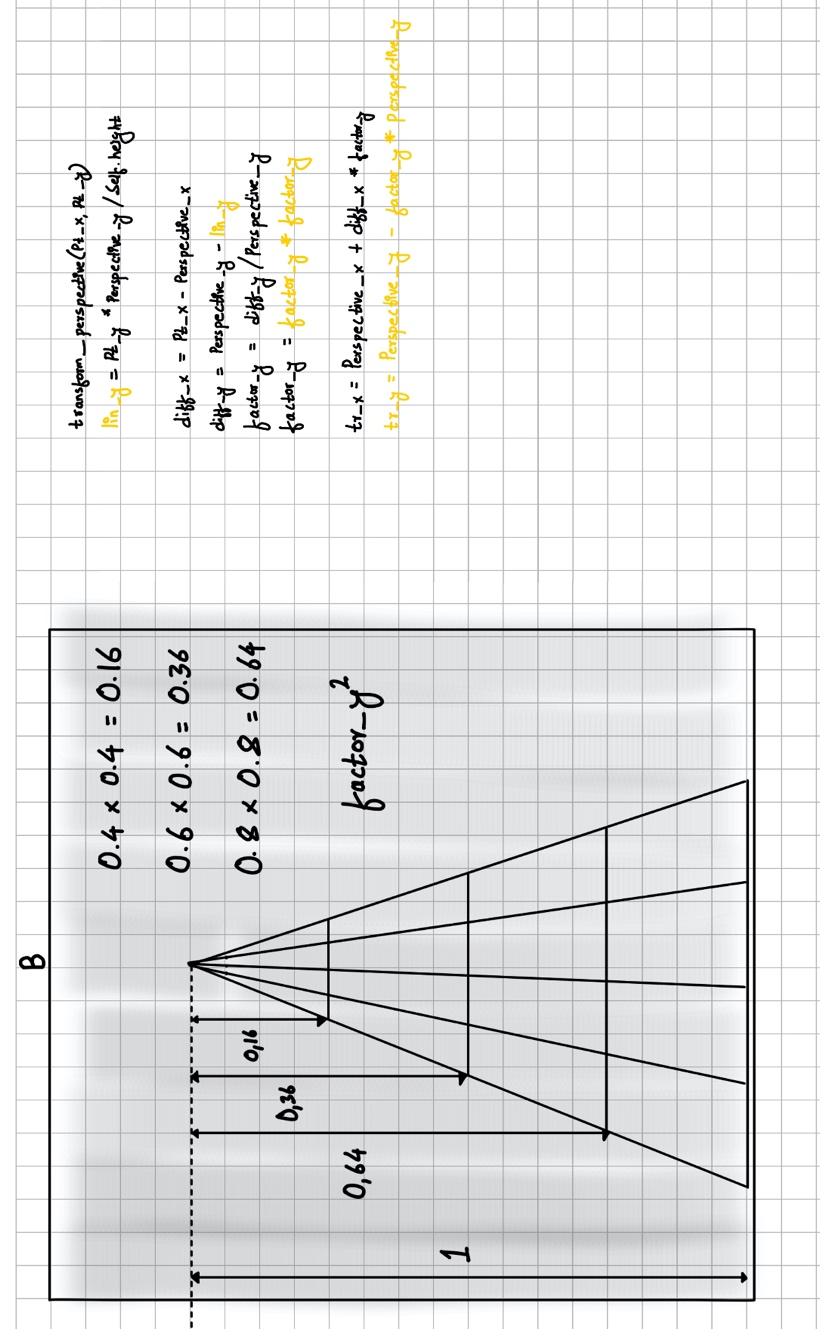
* For the example, we are going to take the blue line.
* Blue line has two points which is (pt\_x1, pt\_y1)
* The Blue line in 2-dimensional space is displayed in a red line, therefore the other point is at the top which is (pt\_x2, pt\_y2).
* These 2D coordinates is what we are going to use for transformation function as an input, i.e., transform\_perspective(pt\_x,pt\_y).
* We are going to compute y transformation. What we want is, when the y is at the maximum value which is (0.height) but we want the maximum of perspective\_y. Therefore, we are going to make proportions.
* For example: If point1 is 0 then we keep 0, if point 2 is height, then we divide by the height and multiply by perspective\_y to be at the perspective\_y point. Hence transformation is: tr\_y = pt\_y \* perspective\_y / self.height
* In the code, we are going to add some condition to avoid going above perspective\_y even if the screen is above the height.
* So after applying the transformation to, y then the point 2 is there in 2D line with red font.
* Now, we are going to compute x and as you can see in the picture, x is going to be closer and closer to the perspective\_x.
* There is diff\_x and diff\_y in the picture. Diff\_x is the same value for poiunt 1 and point 2.
* Hence, diff\_x = pt\_x – perspective\_x
* Concerning diff\_y, we are going to use transform y. So, for the point 1, it is going to be, Perspective\_y – 0 so we have the maximum value here which is perspecrive\_y.
* So, for the point 2, the transform y with point 2 is already there in red font so it’s going to be 0. i.e., perspective\_y – perspective\_y. Hence diff\_y = perspective\_y – tr\_y
* How to compute x, to compute the value of x, we need to understand that the delta of x depends on y. The closer we go to the perspective points vertically, diff\_x will be smaller. And if we go halfway vertically, diff\_x value will be 50%.
* What we want to have is proportion variable, i.e., 1 when diff\_y is at the maximum value till perspective\_y. And is 50% when diff\_y is at the halfway.
* We are going to apply this vertical proportion to the difference of x and we will call that factor\_y, which is: factor\_y = diff\_y / perspective\_y
* Factor\_y equals 1 in the case of point no.1 and it equals 0 In the case of point no. 2. Because we have 0 in factor\_y for the point no. 2 to transform x will not have any difference with perspective\_x. Hence, tr\_x = perspective\_x + diff\_x \* factor\_y
* And finally, we will return the transform coordinates. (return tr\_x, tr\_y)

## In V1 Horizontal Lines Perspective Continued:

A graph of a triangle with lines and arrows

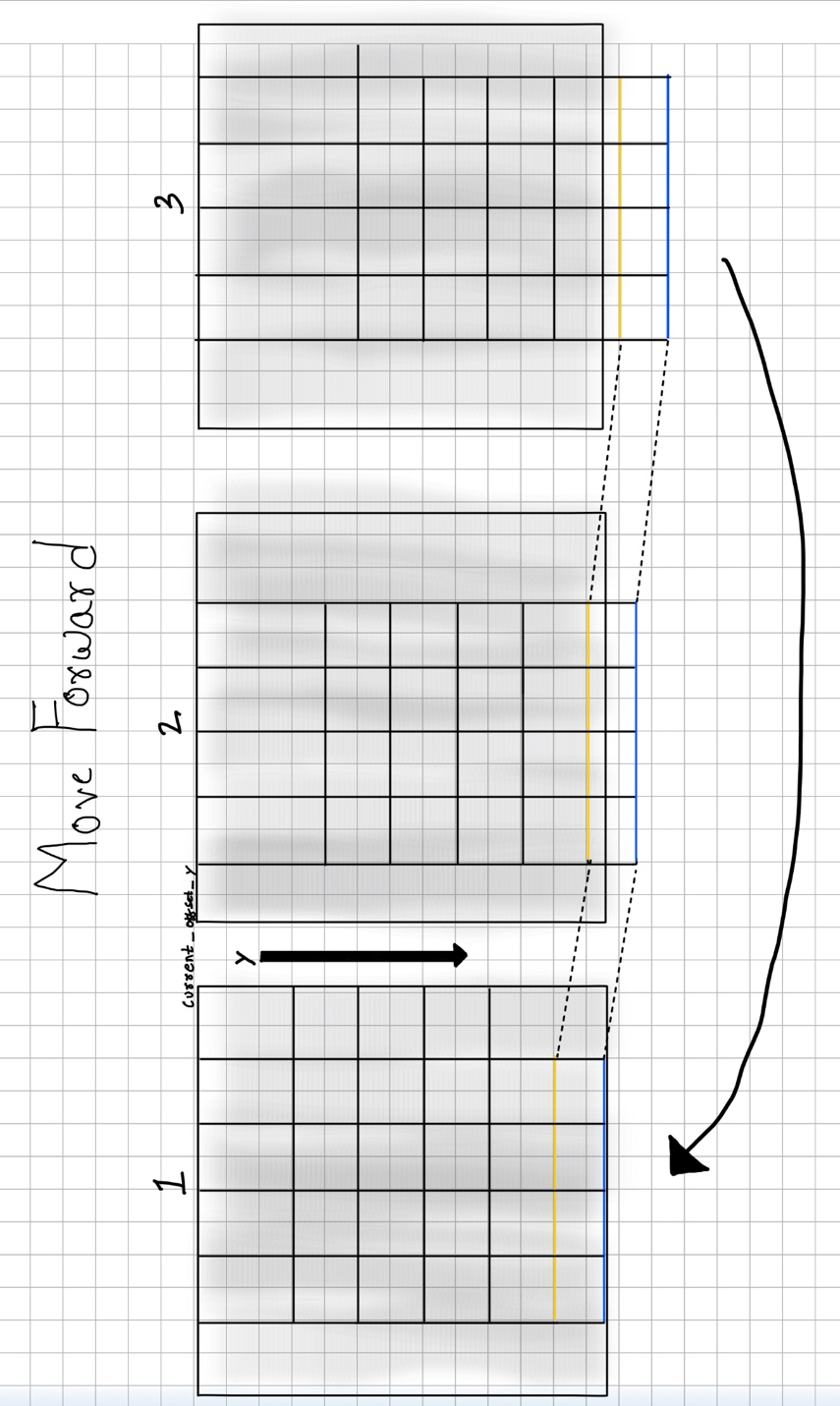
Description automatically generated

* After that, from the above picture, we are going to make Horizontal Lines Perspective.
* We are going to use transformation function to display it into perspective.
* There is a phenomenon where the front tiles are shorter and last back tiles are bit longer. This is because we used a constant spacing between the horizontal lines.
* We are going to change to get different result like picture below: 
* We have kind of attraction effect, the more we go closer, the more we get attracted sort of effect so that the spacing in the front gets bigger and gets smaller at the last.
* As discussed previously, factor\_y is a proportional vertical factor i.e., 100% distance from perspective point and the closer we go to the perspective point then the distance will decrease.
* The main thing is, we must get factor\_y smaller than expected. That means it’s going to go faster to the perspective point.
* The idea is to multiply the factor by itself to get the smooth moving transition sort of effect. Factor\_y2
* This is kind of attraction behaviour, and we are going to apply this to transformation function.
* On the picture below: Yellow are the modifications in the transform function. We renamed the variable as lin\_y because we are going to use that as an intermediate value.



Perspective\_y



* Because the factor has changed, the compute is going to be changed too which means we are going to change the transform perspective function and then at last we are going to return tr\_x, tr\_y.
* Now, we are going to have an animation of moving forward.
* 
* On the update function we have the parameter which is “dt”. And when calling Clock.schedule\_interval(self.update, 1.0 / 60.6) it means, it’s going to be called 1 time and every 1 60th of seconds. It is not going to be as precise as per the Machine it runs. On the old machines, we can use 30 or 50 fps, but the problem is the update function will be called less regularly and the game forwards speed is going to be slower. Resulting the game being easy for an old machine.
* Because the speed will not be as precise as it is because of older machine and ram usages, we don’t want the game to be easy for old machines even we don’t have the exact fps.
* We are going to use delta time (dt) which is a precise value. We are going to use dt twice, i.e., we are going forward twice the time which is proportional to delta time.
* We are going to add twice the speed because we are going to be slower on machines.
* Now to go left in the screen we are using this code: self.current\_offset\_x += self.SPEED\_X \* time\_factor
* Now to go left and right in the screen, we are going to manage the touch and eventually use keyboard to steer left and right.

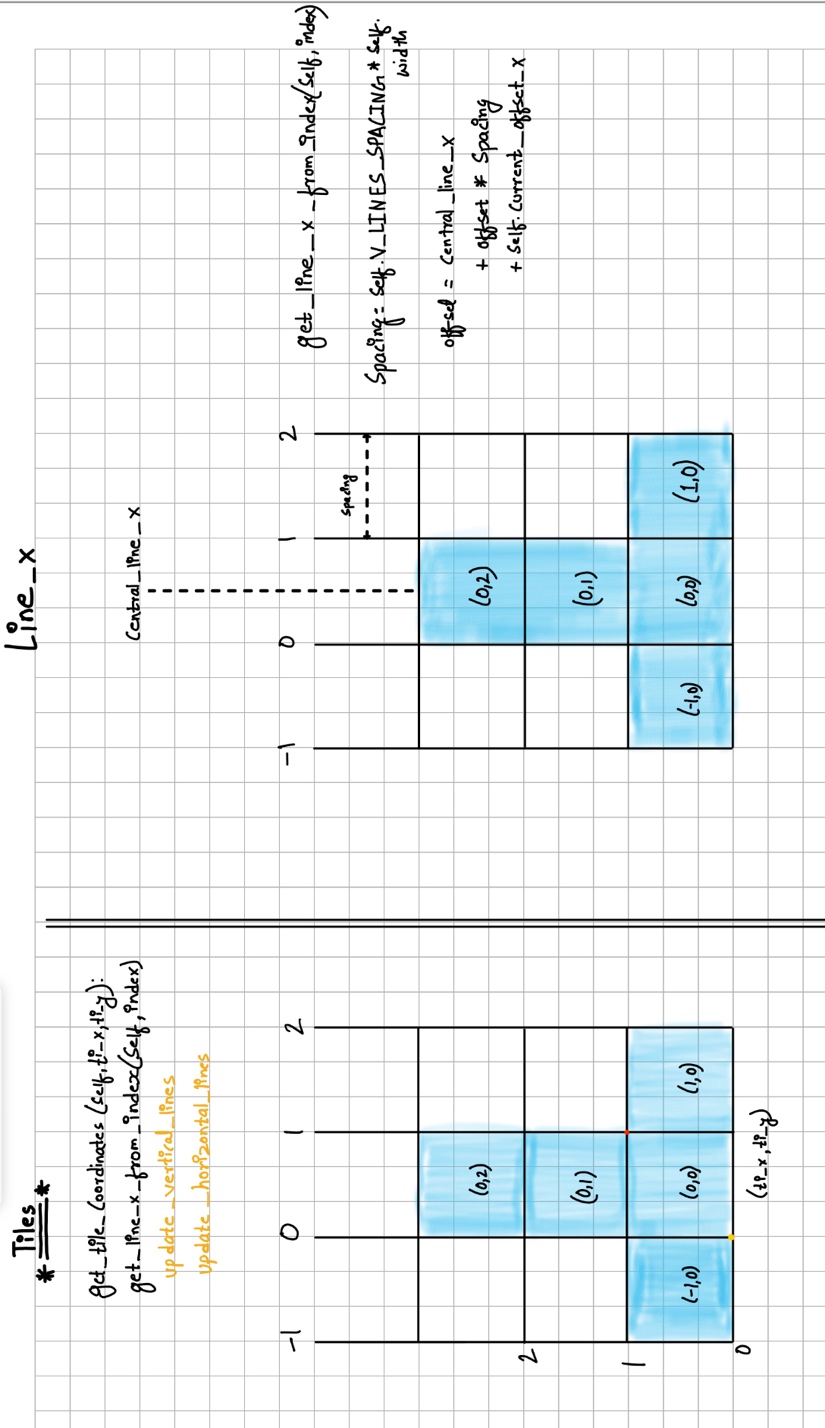
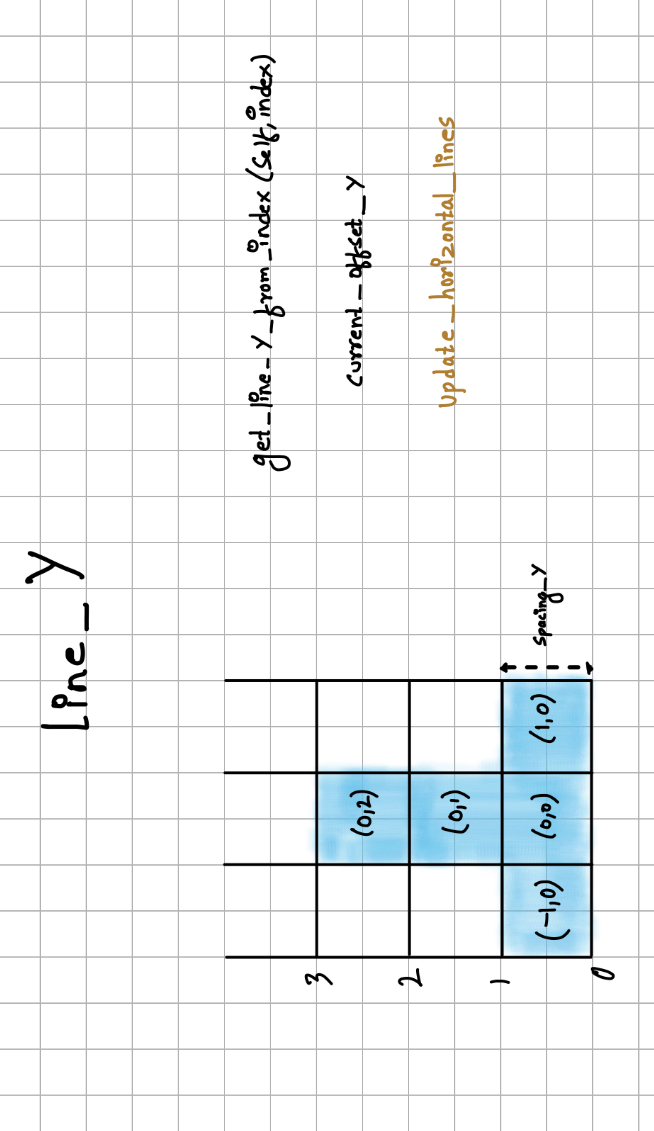
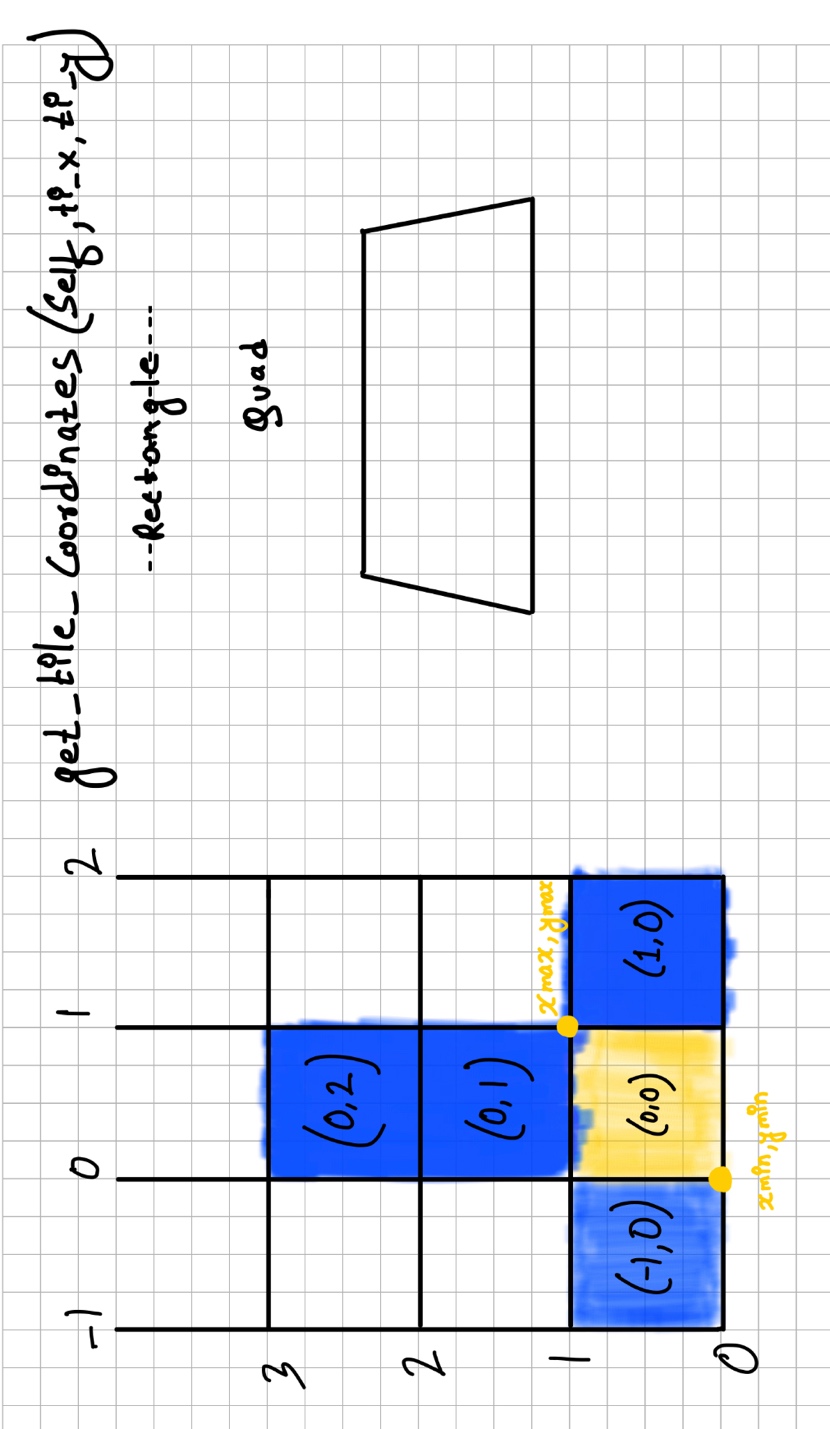
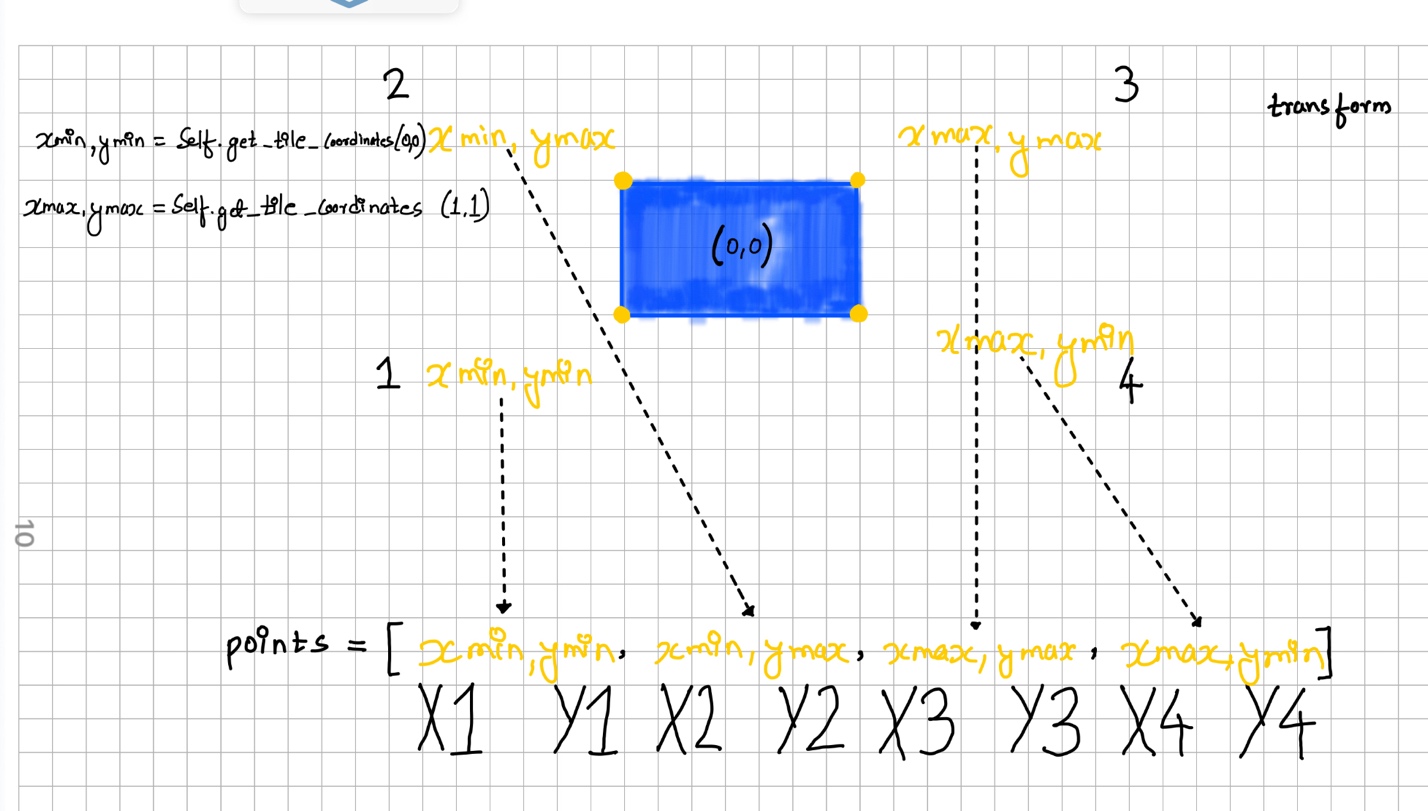
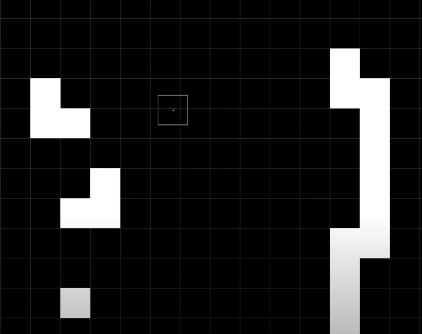
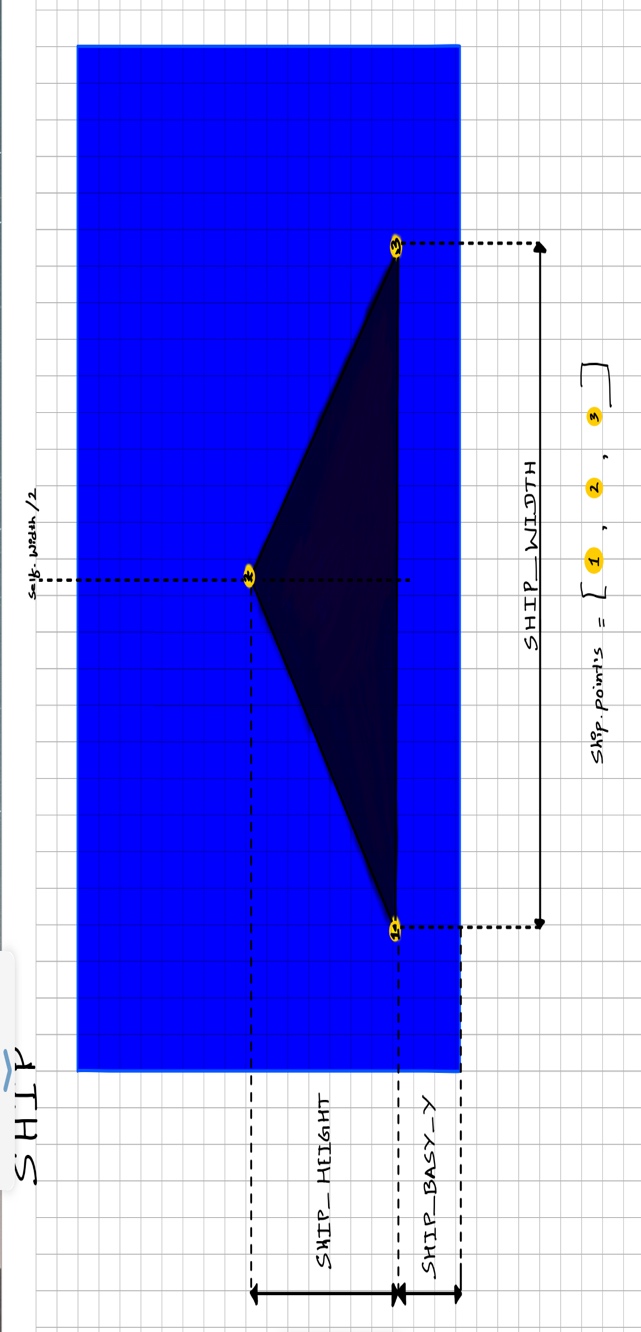
# V2 Widgets:

* In version 2 of making Chandra Yan, we will first make:
* Structure and organise the code
* Display the path (tiles). At first, we will display one tile and then we will display multiples tiles.
* Land generation. We will generate the path for the yan to travel using land generation algorithm.
* We will display the Yan (spaceship).
* We will check if the yan (ship) is still on the track and then if there is a Collisions then we will print (Game Over) but the game will continue at that stage.

## In V2 Continue: Structure the Code:

* So, I created a new python file (trans) inside the Chandra yan project and paste the transform function in it.
* After cut and paste the transform function separately, the project didn’t run and showed the error saying ('MainWidget' object has no attribute 'transform\_perspective') therefore we have commanded the main widget that it has to include the transform function.

## In V2 Continue: Tiles and Line Coordinates:

* From the picture below:
* 
* To display tiles in the game, we are first going to display 4 vertical and 4 horizontal lines.
* Let’s say if we want to display the tile (0,0), (1,0), (-1,0), (0,1), (0,2) which will be the coordinate system. We are going to call the tile coordinates as tile index. i.e. (ti\_x, ti\_y)
* Tile index will be useful to have the function get\_tile\_coordinates(self, ti\_x, ti\_y): that will take the index and return the actual x & y coordinates of the lower left point. (Yellow dot).
* If we have the lower left point, then we can also have the upper right point. If we give the +1x index and +1y index in red dot, then form these coordinates it will be easy to draw the tile.
* To implement the function get\_tile\_coordinates(self, ti\_x, ti\_y): we need to get the coordinates of the different lines. And for that we are going to assign index to those different lines. For example: there are lines which are assigned as -1, 0, 1, 2. (if there are more lines, we add more index as -3, -2, -1 …. etc)
* For the vertical lines, it is same which is line index 0,1,2,3.
* We will get x coordinates of all the vertical lines firstly.
* And then we will write the function get\_line\_x\_from\_index(self, index). For example: If we put index 0, in the function then we are going to get x coordinate of this (0) line likewise if we put index 2 then we will get x coordinate of line 2. After we get the x coordinates of vertical lines (indexes) then we are going to apply the same function to get the coordinates of the horizontal lines. This process will make easy to implement get\_tile\_coordinates(self, ti\_x, ti\_y): function.
* After that we are going to use the function get\_line\_x\_from\_index(self, index) to put code inside update\_vertical lines and update\_horizontal\_lines.
* So, the process to implement the new function i.e., get\_line\_x\_from\_index(self, index) are:
* As we have central\_line\_x and spacing between them. Spacing is spacing = self.V\_LINES\_SPACING \* self.width.
* If there is an index of 0, we can see it is not aligned with centre\_line\_x (center of the screen) i.e. it is half way to the left. Therefore we will compute an offset which is offset = index – 0.5 so the line\_x will be line\_x = central\_line\_x + offset \* spacing + self.current\_offset\_x. for example: if we have a index of -1 then we will have an off\_set of -1.5 and if we start from central\_line\_x then we are at -1.5 and spacing and we need to add self.current\_offset\_x which is the global offset when we press left and right keys.
* Now let’s check the picture:
* 
* We are going to implement this function: get\_line\_y\_from\_index(self, index). If we give index 0, it’s going to return 0. If we give index 1 then it’s going to be spacing\_y. Index 2 is going to be y value for line 2. Etc.
* We should not forget to use current\_offset\_y.
* We are also going to rework on update\_horizontal\_lines function to use the new function get\_line\_y\_from\_index(self, index).
* Now lets display the tiles.
* Lets suppose if we want to display the tile 0(0,0), firstly we need to implement get\_tile\_coordinates(self, ti\_x, ti\_y): function in order to get lower left coordinate which is xmin, ymin. If we pass the coordinates.
* Now we are going to display the tile:
* 
* We are going to display the first tile 0 (0,0) and for that we are going to use the function get\_tile\_coordinates(self, ti\_x, ti\_y) to get the lower left coordinates which is the yellow dot xmin, ymin.
* When we pass that function +1 and +1 of width and height, we can get x max and y max coordinate, making it easy to display the tile. We are not building the rectangle; in perspective mode we are going to develop something like above right on the picture.
* We need to give 4 coordinates of that perspective design of spaceship. For that we are going to use quad not rectangle because rectangle only use width and height and quad gives un 4 points.
* So how does the quad works?
* 
* So we have x min and y min that we get from xmin,ymin = self.get\_tile\_coordinates(0,0) and we get x max, y max from +1 and +1. (Width and height of the tile) i.e., xmax, ymax = self.get\_tile\_coordinates(1,1) and likewise we have other two points which is xmin, ymax and xmax, ymin. We need to give these 4 points to the quad. So, the point will get the point list. In which order we need to give the points is up to us, but we need to rotate in a way or other way. We can’t go criss cross.
* So, we need to take a point and rotate, in this case, we are going to use point 1,2,3 & 4. Therefore list of points = [1,2,3,4] i.e., points = [ xmin, ymin, xmin, ymax, xmax, ymax, xmax, ymin]
* And we need to call the transform function at the last moment which means we need to use intermediate variable which are x1, y1 after transformed from xmin, ymin and so on for all the other coordinates.
* To go make the lands generate left and right we are going to have a random number to generate the tiles in three patters which is below:
* 
* After the land is generated, now we are going to make a spaceship (Triangle).
* 
* We are going to place the triangle in the bottom central horizontally of the screen which is self.width / 2
* We are going to define a variable SHIP\_WIDTH to define the width of the ship with a percentage of the screen width also SHIP\_HEIGHT and SHIP\_BASE\_Y to define the height and base of the ship. SHIP\_BASE\_Y is for the base so that the ship is not totally on the bottom of the screen.
* Concerning the canvas, we are going to use triangle instructions. For that we are going to need three points, i.e., Point 1, Point 2 and point 3.
* For this process, we are going to implement the three point coordinates for the ship and add it to the ship variable which is variable for ship points, ship.points = [ 1, 2, 3].

# V3 Canvas:

* We are going to have a menu to play the game and go to the game over screen when the ship is collided. We also display the score.
* We are going to manage the game states where Start and Game over will be displayed.
* We are going to make the button to start the game.
* We are going to make Menu screen before starting the game and after the game finish.
* At last, we are going to finalize the game showing the background, sounds and score.

## In V3 canvas -> Game Over state

* We are going to add a state\_game\_over variable and show the game over when the ship is collided.

Finally, after adding the images and sound effects we are now able to launch the game.

# Tools and Sources Used:

* Thanks to Microsoft Bing to create a manipulated photo for the background of the game.
* Logic Pro X software to create a special sound effect.
* Special Thanks to [Jonathan Roux](https://www.linkedin.com/in/rouxjonathan) for creating the tutorial video to learn python and kivy and create the game.