CCE60220

Perangkat Bergerak (TKOM)

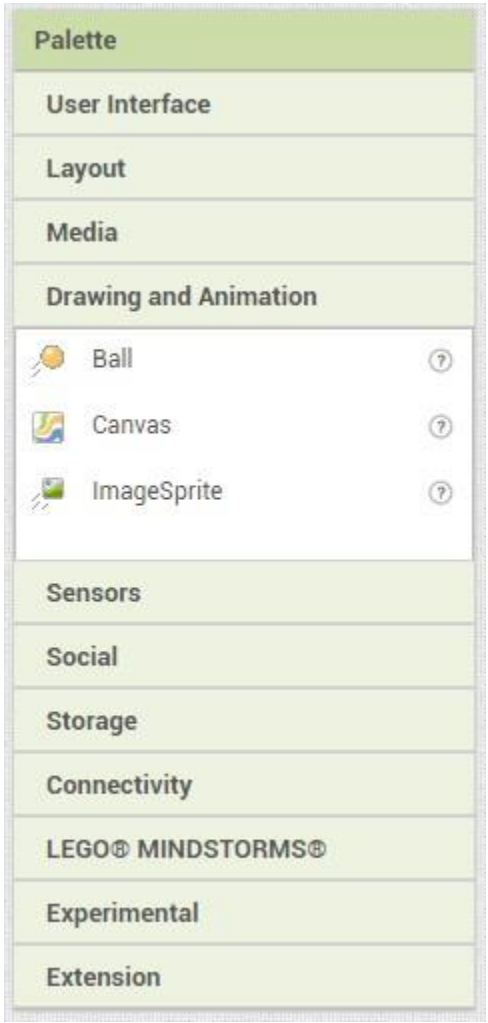
Fakultas Ilmu Komputer Universitas Brawijaya

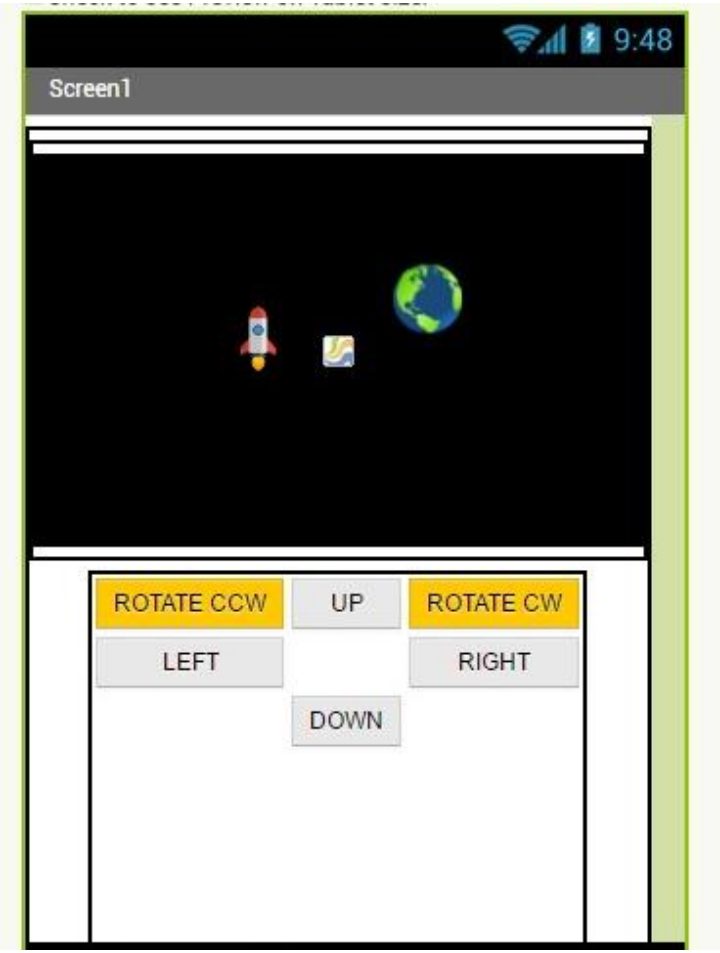
Agenda Perkuliahan

1. ~~Intro dan overview perkuliahan~~
2. ~~Sejarah dan perkembangan teknologi perangkat bergerak~~
3. ~~Komponen perangkat keras dan perangkat lunak~~
4. ~~Pengenalan dan instalasi android studio serta aplikasi sederhana~~
5. ~~Intent dan passing data pada Android Studio~~
6. ~~Android Studio: Sensor reading~~
7. ~~Android Studio: Storage & shared preference~~
8. **==========UTS**
9. ~~Pengenalan dan aplikasi sederhana dengan MIT AppInventor~~
10. ~~Appinventor: variable, conditional, tinyDB, file~~
11. ~~appInventor: sensor reading & persiapan project~~
12. ~~Appinventor: Akuisisi gambar dan suara~~
13. ~~Appinventor: komunikasi bluetooth, Wifi to control device (http get)~~
14. Appinventor: basic animation and using procedures
15. Presentasi kelompok
16. **==========UAS**

Using basic animation and using procedures

Basic of animation





Basic of animation

Canvas

- A two-dimensional touch-sensitive rectangular panel on which drawing can be done and sprites can be moved.
- The BackgroundColor, PaintColor, BackgroundImage, Width, and Height of the Canvas can be set in either the Designer or in the Blocks Editor.
- The Width and Height are measured in pixels and must be positive.
- Any location on the Canvas can be specified as a pair of (X, Y) values, where
  - X is the number of pixels away from the left edge of the Canvas
  - Y is the number of pixels away from the top edge of the Canvas

Canvas: coordinates

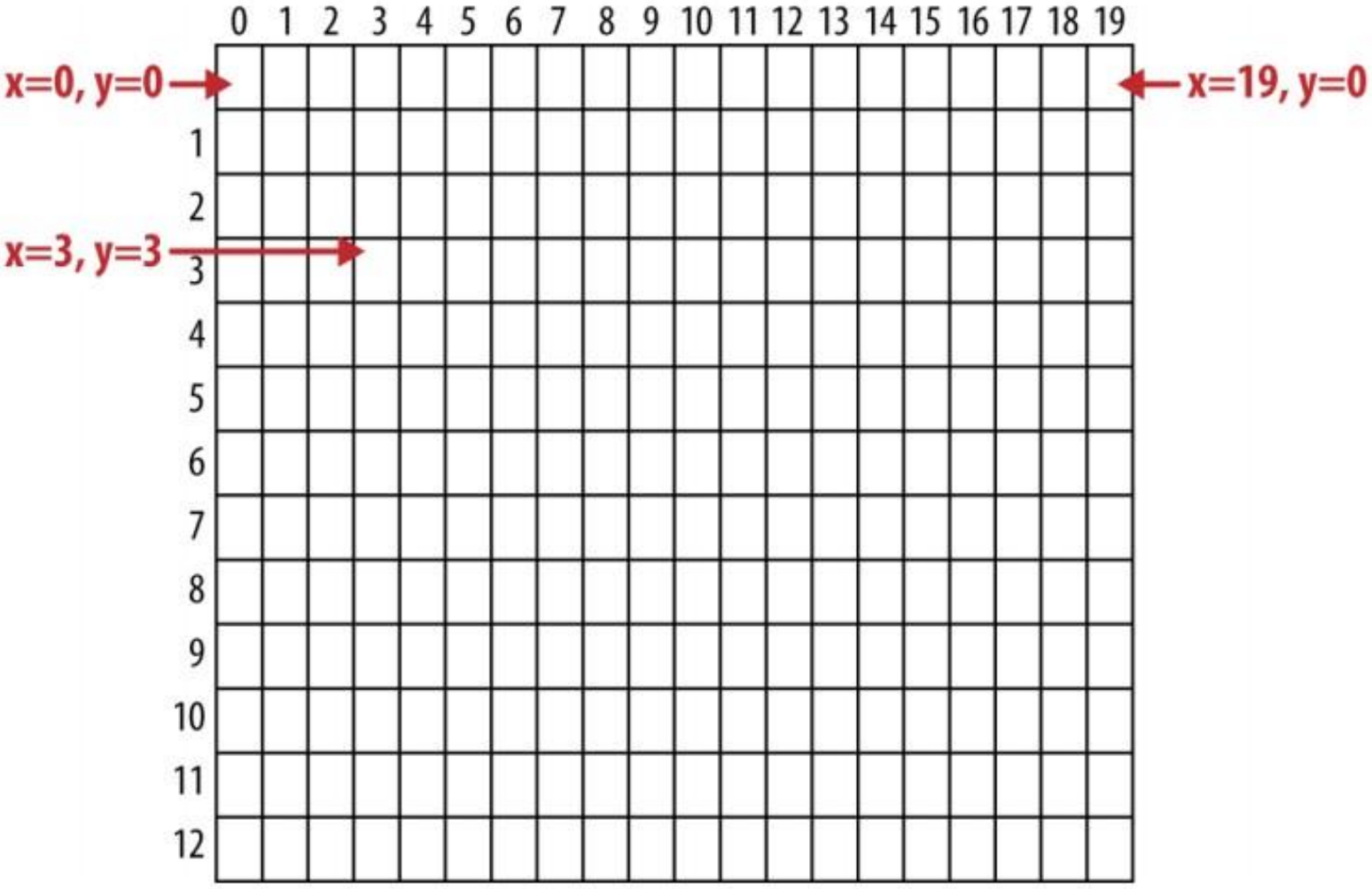


Figure 17-2. The Canvas coordinate system

Canvas

Properties

- BackgroundColor: The color of the canvas background.
- BackgroundImage: The name of a file containing the background image for the canvas
- FontSize: The font size of text drawn on the canvas.
- Height
- LineWidth: The width of lines drawn on the canvas.
- PaintColor: The color in which lines are drawn
- TextAlignment: Determines the alignment of the text drawn by DrawText() or DrawAngle() with respect to the point specified by that command: point at the left of the text, point at the center of the text, or point at the right of the text.
- Visible: Specifies whether the component should be visible on the screen. Value is true if the component is showing and false if hidden.
- Width

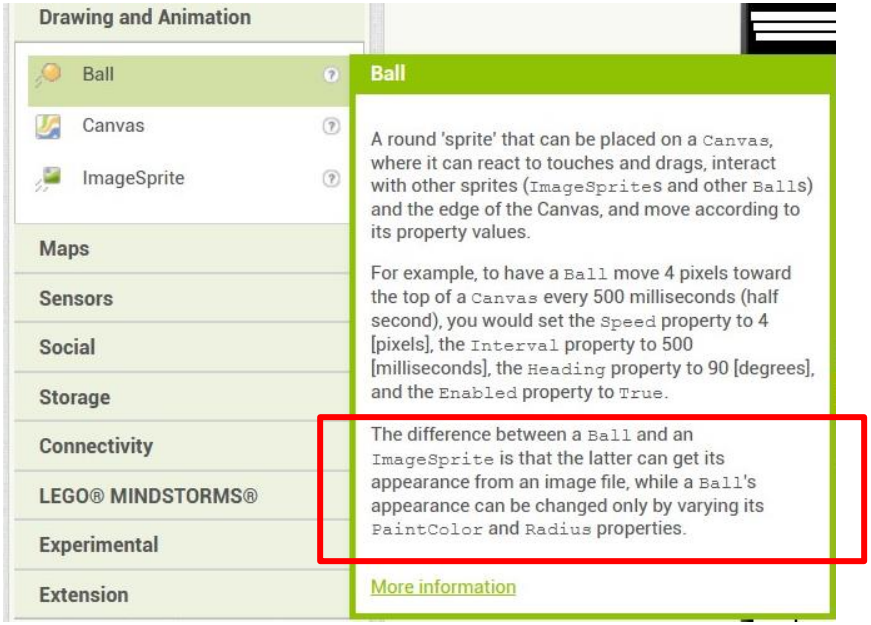
Events

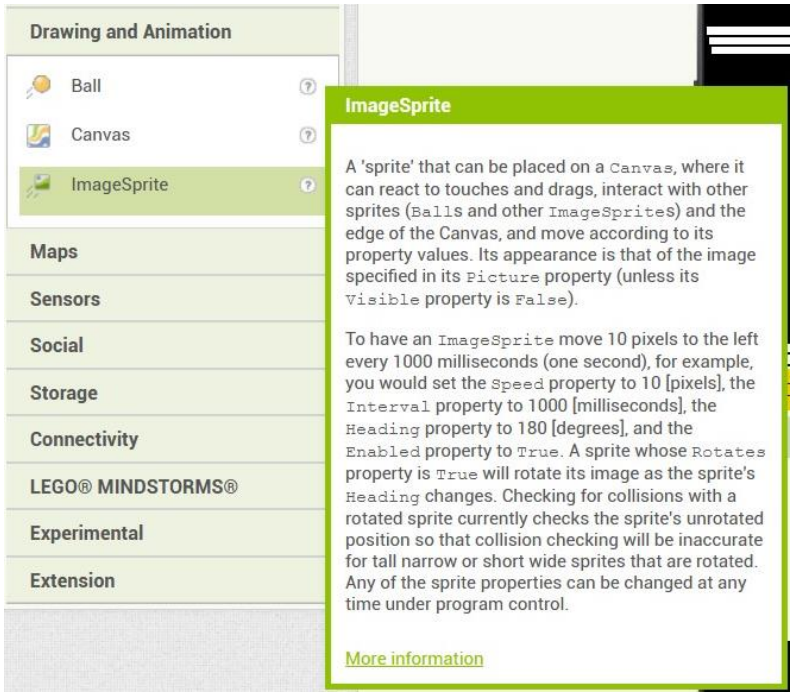
- **Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY, boolean draggedSprite):** When the user does a drag from one point (prevX, prevY) to another (x, y). The pair (startX, startY) indicates where the user first touched the screen, and "draggedSprite" indicates whether a sprite is being dragged.
- **Flung(number x, number y, number speed, number heading, number xvel, number yvel, boolean flungSprite):** When a fling gesture (quick swipe) is made on the canvas: provides the (x,y) position of the start of the fling, relative to the upper left of the canvas. Also provides the speed (pixels per millisecond) and heading (0-360 degrees) of the fling, as well as the x velocity and y velocity components of the fling's vector. The value "flungSprite" is true if a sprite was located near the the starting point of the fling gesture.
- **TouchDown(number x, number y):** When the user begins touching the canvas (places finger on canvas and leaves it there): provides the (x,y) position of the touch, relative to the upper left of the canvas
- **TouchUp(number x, number y):** When the user stops touching the canvas (lifts finger after a TouchDown event): provides the (x,y) position of the touch, relative to the upper left of the canvas
- **Touched(number x, number y, boolean touchedSprite):** When the user touches the canvas and then immediately lifts finger: provides the (x,y) position of the touch, relative to the upper left of the canvas. TouchedSprite is true if the same touch also touched a sprite, and false otherwise.

Methods

- **Clear():** Clears anything drawn on this Canvas but not any background color or image.
- **DrawCircle(number x, number y, number r):** Draws a circle (filled in) at the given coordinates on the canvas, with the given radius.
- **DrawLine(number x1, number y1, number x2, number y2):** Draws a line between the given coordinates on the canvas.
- **DrawPoint(number x, number y):** Draws a point at the given coordinates on the canvas.
- **DrawText(text text, number x, number y):** Draws the specified text relative to the specified coordinates using the values of the FontSize and TextAlignment properties.
- **DrawTextAtAngle(text text, number x, number y, number angle):** Draws the specified text starting at the specified coordinates at the specified angle using the values of the FontSize and TextAlignment properties.
- **number GetBackgroundPixelColor(number x, number y):** Gets the color of the specified point. This includes the background and any drawn points, lines, or circles but not sprites.
- **number GetPixelColor(number x, number y):** Gets the color of the specified point.
- **text Save():** Saves a picture of this Canvas to the device's external storage. If an error occurs, the Screen's ErrorOccurred event will be called.
- **text SaveAs(text fileName):** Saves a picture of this Canvas to the device's external storage in the file named fileName. fileName must end with one of .jpg, .jpeg, or .png, which determines the file type.
- **SetBackgroundPixelColor(number x, number y, number color):** Sets the color of the specified point. This differs from DrawPoint by having an argument for color.

sprite


Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Sensors
Social
Storage
Connectivity
LEGO® MINDSTORMS®
Experimental
Extension
Ball
A round 'sprite' that can be placed on a Canvas, where it can react to touches and drags, interact with other sprites (ImageSprites and other Balls) and the edge of the Canvas, and move according to its property values.
For example, to have a Ball move 4 pixels toward the top of a Canvas every 500 milliseconds (half second), you would set the Speed property to 4 [pixels], the Interval property to 500 [milliseconds], the Heading property to 90 [degrees], and the Enabled property to True.
The difference between a Ball and an ImageSprite is that the latter can get its appearance from an image file, while a Ball's appearance can be changed only by varying its PaintColor and Radius properties.
More information


Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Sensors
Social
Storage
Connectivity
LEGO® MINDSTORMS®
Experimental
Extension
ImageSprite
A 'sprite' that can be placed on a Canvas, where it can react to touches and drags, interact with other sprites (Balls and other ImageSprites) and the edge of the Canvas, and move according to its property values. Its appearance is that of the image specified in its Picture property (unless its Visible property is False).
To have an ImageSprite move 10 pixels to the left every 1000 milliseconds (one second), for example, you would set the Speed property to 10 [pixels], the Interval property to 1000 [milliseconds], the Heading property to 180 [degrees], and the Enabled property to True. A sprite whose Rotates property is True will rotate its image as the sprite's Heading changes. Checking for collisions with a rotated sprite currently checks the sprite's unrotated position so that collision checking will be inaccurate for tall narrow or short wide sprites that are rotated. Any of the sprite properties can be changed at any time under program control.
More information

ImageSprite

A 'sprite' that can be placed on a Canvas, where it can react to touches and drags, interact with other sprites (Balls and other ImageSprites) and the edge of the Canvas, and move according to its property values. Its appearance is that of the image specified in its Picture property (unless its Visible property is False.

A sprite whose Rotates property is True will rotate its image as the sprite's Heading changes. Checking for collisions with a rotated sprite currently checks the sprite's unrotated position so that collision checking will be inaccurate for tall narrow or short wide sprites that are rotated. Any of the sprite properties can be changed at any time under program control.

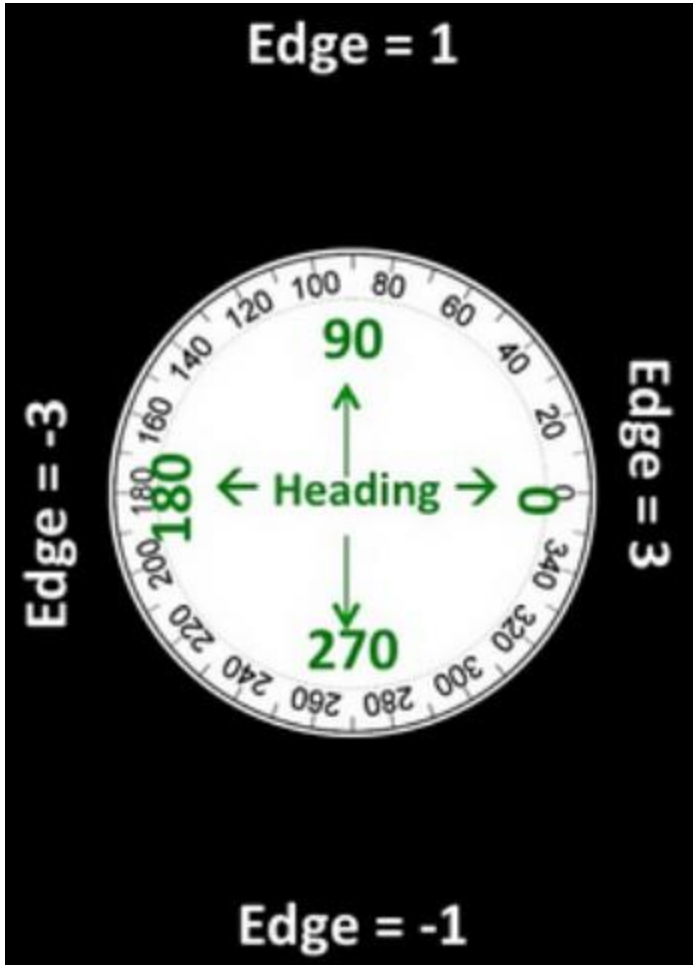


Image sprite

Properties

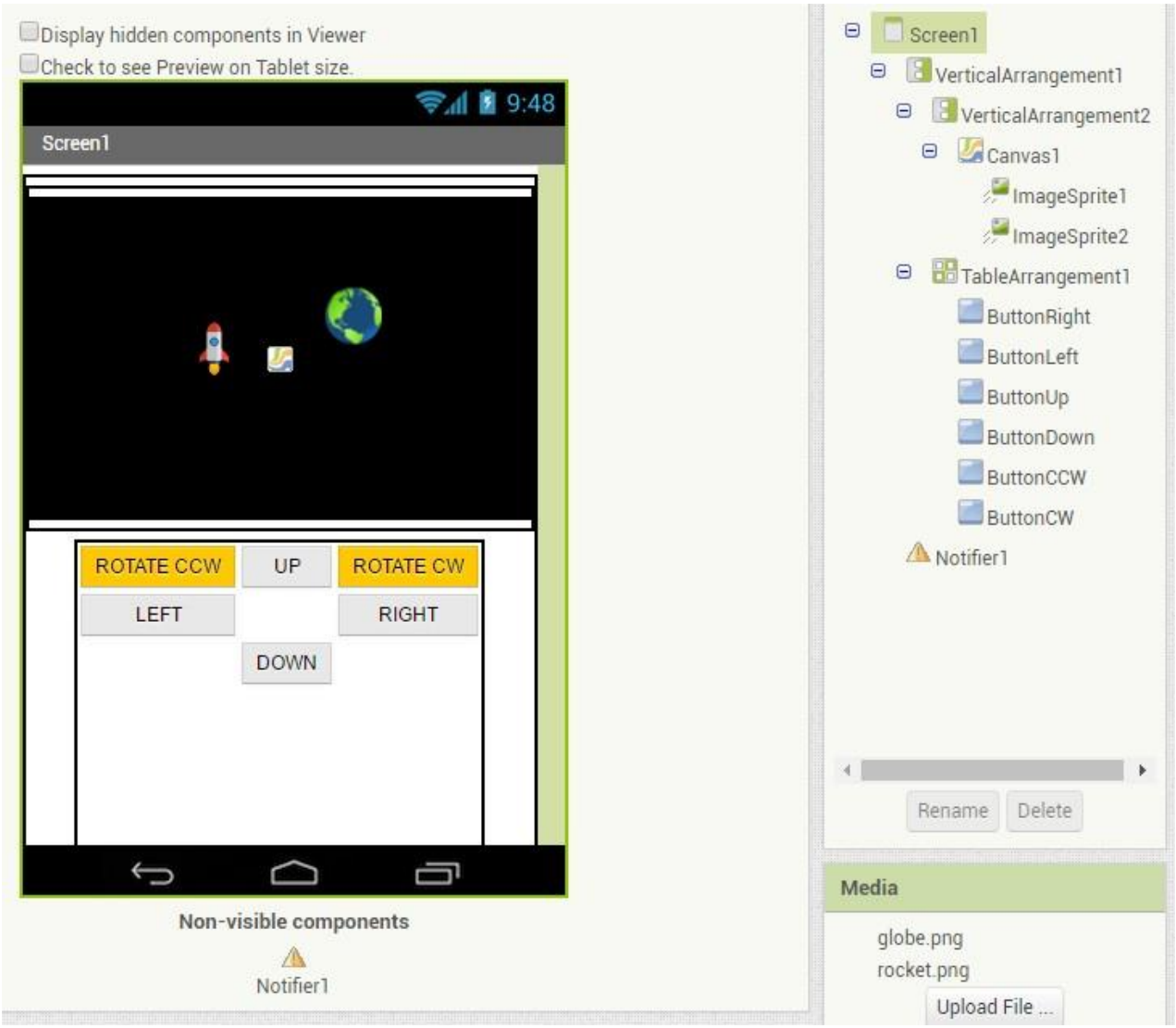
- Enabled: Controls whether the sprite moves when its speed is non-zero.
- Heading: Returns the sprite's heading in degrees above the positive x-axis. Zero degrees is toward the right of the screen; 90 degrees is toward the top of the screen.
- Height
- Interval: The interval in milliseconds at which the sprite's position is updated. For example, if the interval is 50 and the speed is 10, then the sprite will move 10 pixels every 50 milliseconds.
- Picture: The picture that determines the sprite's appearance
- Rotates: If true, the sprite image rotates to match the sprite's heading. If false, the sprite image does not rotate when the sprite changes heading. The sprite rotates around its CenterPoint.
- Speed: The speed at which the sprite moves. The sprite moves this many pixels every interval.
- Visible: True if the sprite is visible.
- Width
- X: The horizontal coordinate of the left edge of the sprite, increasing as the sprite moves to the right.
- Y: The vertical coordinate of the top of the sprite, increasing as the sprite moves down.
- Z: How the sprite should be layered relative to other sprits, with higher-numbered layers in front of lower-numbered layers.

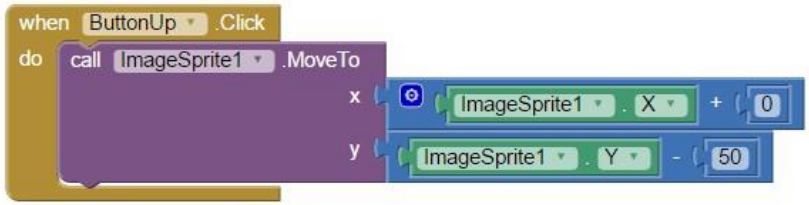
Events

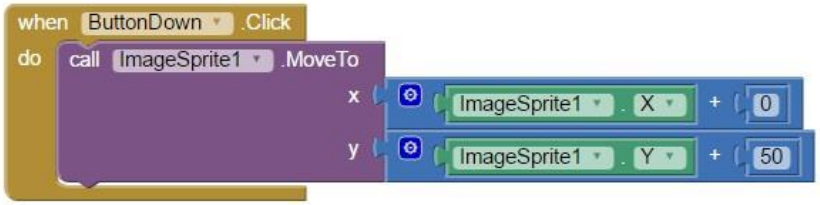
- **CollidedWith(component other):** Handler for CollidedWith events, called when two sprites collide. Note that checking for collisions with a rotated ImageSprite currently checks against the sprite's unrotated position. Therefore, collision checking will be inaccurate for tall narrow or short wide sprites that are rotated.
- **Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY):** Handler for Dragged events. On all calls, the starting coordinates are where the screen was first touched, and the "current" coordinates describe the endpoint of the current line segment. On the first call within a given drag, the "previous" coordinates are the same as the starting coordinates; subsequently, they are the "current" coordinates from the prior call. Note that the Sprite won't actually move anywhere in response to the Dragged event unless MoveTo is specifically called.
- **EdgeReached(number edge):** Event handler called when the sprite reaches an edge of the screen. If Bounce is then called with that edge, the sprite will appear to bounce off of the edge it reached. Edge here is represented as an integer that indicates one of eight directions north(1), northeast(2), east(3), southeast(4), south (-1), southwest(-2), west(-3), and northwest(-4).
- **Flung(number x, number y, number speed, number heading, number xvel, number yvel):** When a fling gesture (quick swipe) is made on the sprite: provides the (x,y) position of the start of the fling, relative to the upper left of the canvas. Also provides the speed (pixels per millisecond) and heading (0-360 degrees) of the fling, as well as the x velocity and y velocity components of the fling's vector.
- **NoLongerCollidingWith(component other):** Event indicating that a pair of sprites are no longer colliding.
- **TouchDown(number x, number y):** When the user begins touching the sprite (places finger on sprite and leaves it there): provides the (x,y) position of the touch, relative to the upper left of the canvas
- **TouchUp(number x, number y):** When the user stops touching the sprite (lifts finger after a TouchDown event): provides the (x,y) position of the touch, relative to the upper left of the canvas
- **Touched(number x, number y):** When the user touches the sprite and then immediately lifts finger: provides the (x,y) position of the touch, relative to the upper left of the canvas

Methods

- **Bounce(number edge):** Makes this sprite bounce, as if off a wall. For normal bouncing, the edge argument should be the one returned by EdgeReached.
- **boolean CollidingWith(component other):** Indicates whether a collision has been registered between this sprite and the passed sprite.
- **MoveIntoBounds():** Moves the sprite back in bounds if part of it extends out of bounds, having no effect otherwise. If the sprite is too wide to fit on the canvas, this aligns the left side of the sprite with the left side of the canvas. If the sprite is too tall to fit on the canvas, this aligns the top side of the sprite with the top side of the canvas.
- **MoveTo(number x, number y):** Moves the sprite so that its left top corner is at the specfied x and y coordinates.
- **PointInDirection(number x, number y):** Turns the sprite to point towards the point with coordinates as (x, y).
- **PointTowards(component target):** Turns the sprite to point towards a designated target sprite. The new heading will be parallel to the line joining the centerpoints of the two sprites.

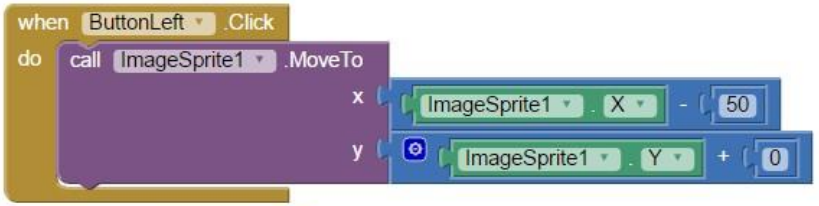

Display hidden components in Viewer
Check to see Preview on Tablet size.
9:48
Screen1
ROTATE CCW
UP
ROTATE CW
LEFT
RIGHT
DOWN
Non-visible components
Notifier1
Screen1
VerticalArrangement1
VerticalArrangement2
Canvas1
ImageSprite1
ImageSprite2
TableArrangement1
ButtonRight
ButtonLeft
ButtonUp
ButtonDown
ButtonCCW
ButtonCW
Notifier1
Rename
Delete
Media
globe.png
rocket.png
Upload File ...

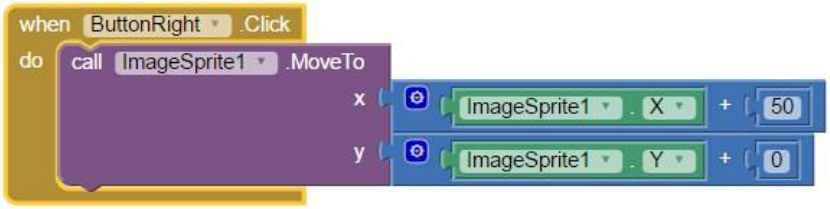

when ButtonUp .Click
do call ImageSprite1 .MoveTo
x ImageSprite1 . X + 0
y ImageSprite1 . Y - 50


when ButtonDown .Click
do call ImageSprite1 .MoveTo
x ImageSprite1 . X + 0
y ImageSprite1 . Y + 50


when ButtonCW .Click
do set ImageSprite1 . Heading to ImageSprite1 . Heading - 10


when ButtonCCW .Click
do set ImageSprite1 . Heading to ImageSprite1 . Heading + 10

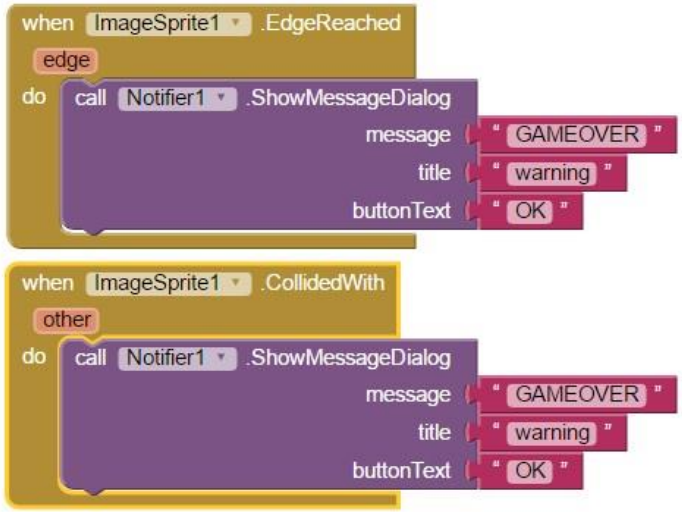

when ButtonLeft .Click
do call ImageSprite1 .MoveTo
x ImageSprite1 . X - 50
y ImageSprite1 . Y + 0


when ButtonRight .Click
do call ImageSprite1 .MoveTo
x ImageSprite1 . X + 50
y ImageSprite1 . Y + 0

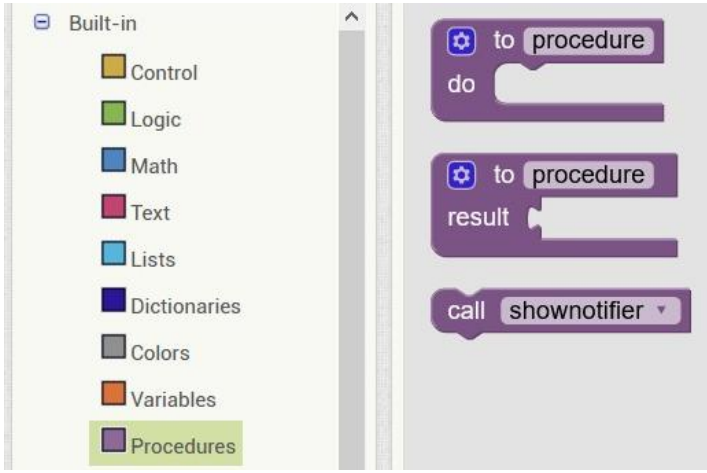

when ImageSprite1 .EdgeReached
edge
do call Notifier1 .ShowMessageDialog
message " GAMEOVER "
title " warning "
buttonText " OK "


when ImageSprite1 .CollidedWith
other
do call Notifier1 .ShowMessageDialog
message " GAMEOVER "
title " warning "
buttonText " OK "

Using procedures: eliminating redundancy

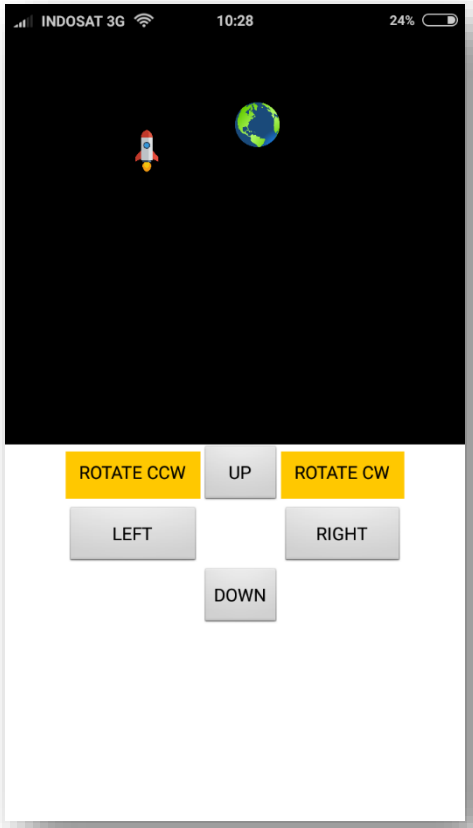


Same function

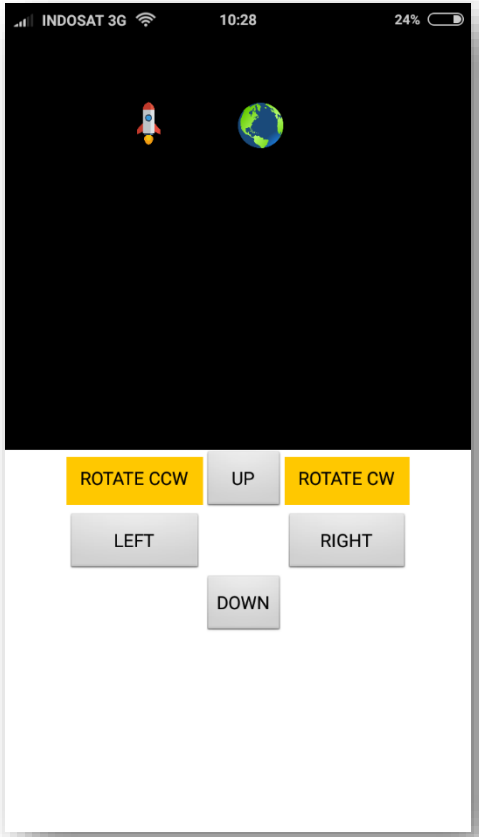


Using procedure and calling procedure:

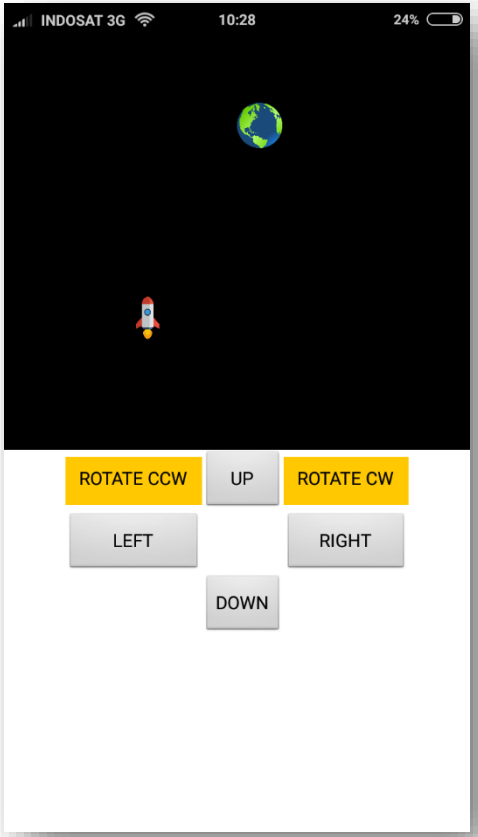
Result



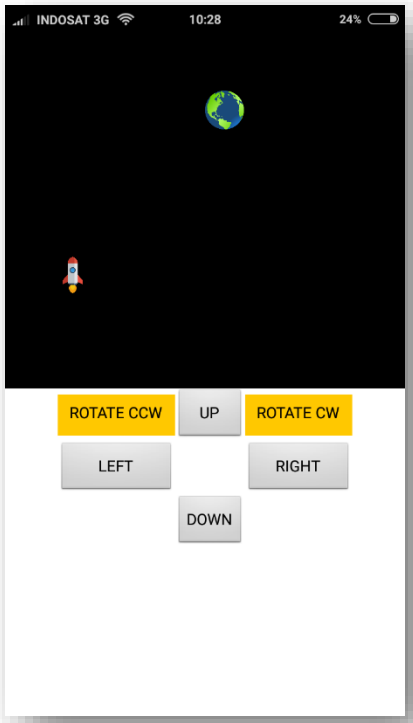
initial



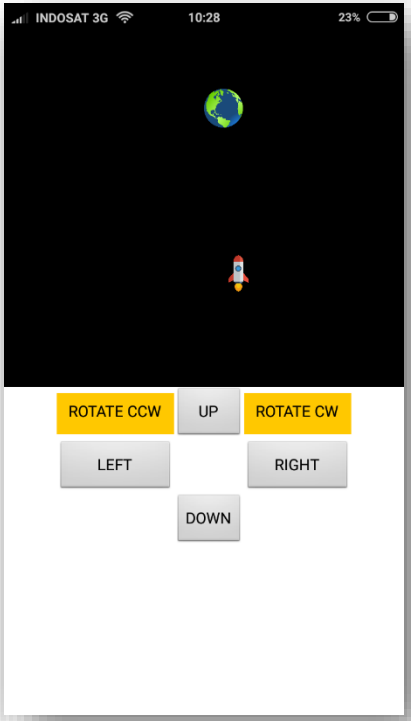
up



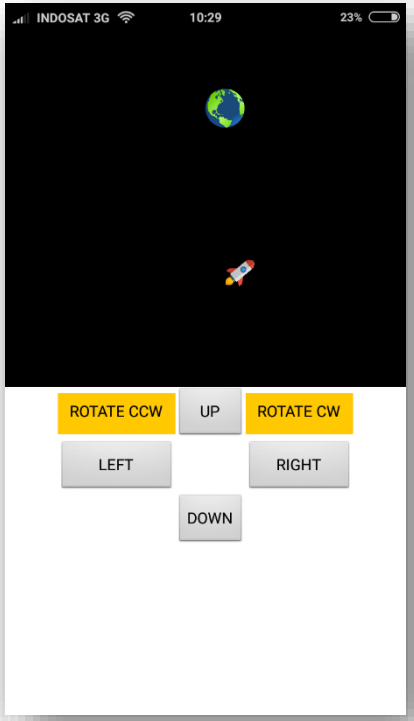
down



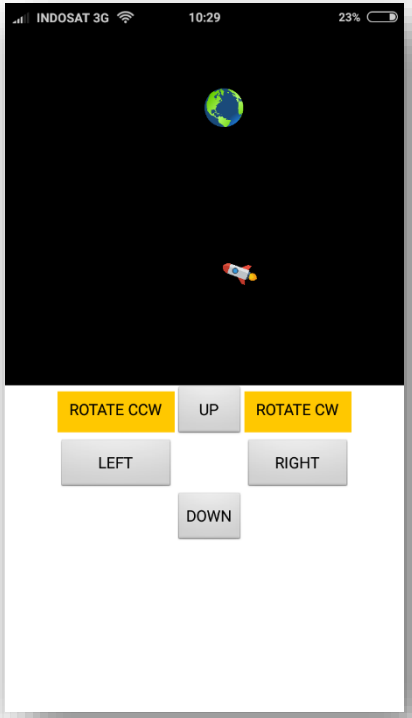
left



right

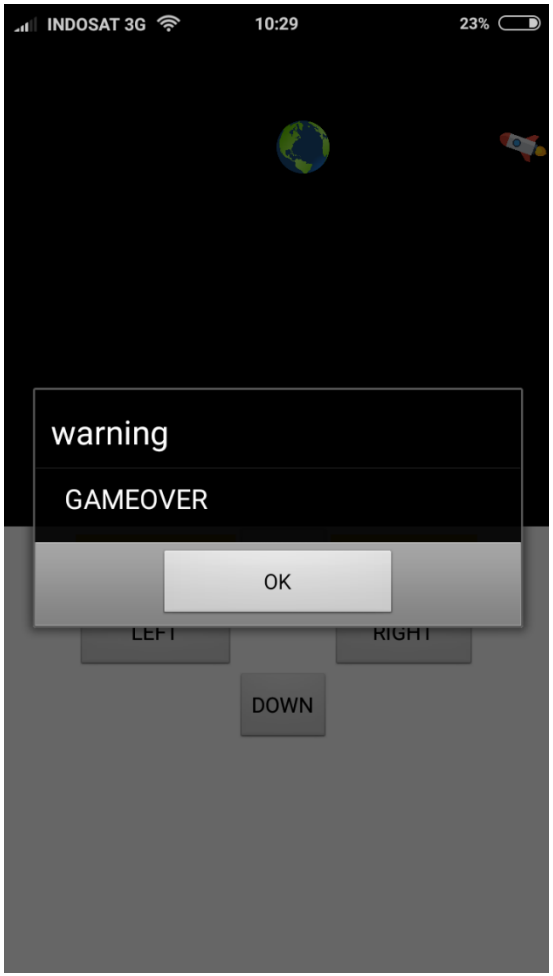


Rotate CW



Rotate CCW

Collide other sprite



Reach edge

TERIMA KASIH