Oblivion:

Introduction: Oblivion is the first program that I have written using the Android Library. Due to the lack of experience I have had with writing programs, the code does not contain documentation in with the code. Oblivion is currently a two-dimensional platform engine for android devices.

Documentation:

* Android Engine:
  + Android Graphics:
    - Android Components:
      * **Coordinates:**
        + Holds a set of coordinates
        + Variables:

double xPos

double yPos

* + - * + Methods:

setX(double x): sets x coordinate.

setY(double y): sets y coordinate.

getX(): returns x coordinate.

getY(): returns y coordinate.

* + - * **GraphicAnimation (inactive):**
        + Takes in a sprite sheet and organizes it to be easily accessible. Divides the sprite sheet and returns a single frame when called.
        + Builder Class:

Constructor

Builder(bitmap image)

Setter Methods:

setWidth(int w)

setHeight(int h)

setNumberOfColumns(int num)

setNumberOfFrames(int num)

* + - * + Variables:

Bitmap spriteSheet: entire spritesheet.

Bitmap frame: current frame.

Int currentFrame: current frame number.

Int frameHeight: height for single frame (px).

Int frameWidth: width for single frame (px).

Int frameColumn: number of frames per column

Int numberOfFrames: total number of frames

* + - * + Constructor:

GraphicAnimation(Builder builder): Creates GraphicAnimation from Builder

GraphicAnimation(Bitmap s, int w, int h, int num, int column): Creates GraphicAnimation from data.

GraphicAnimation(GraphicAnimation ga): clones existing GraphicAnimation

* + - * + Methods:

void setFrame(int frameNumber): Sets the current frame to the given frame number

void nextFrame(): changes the currentFrame to next frame.

int getHeight(): returns width of frame

int getWidth(): returns height of frame

Bitmap getFrame(): returns the currentFrame

* + - * **GraphicBackground**:
        + Creates a panel to hold and manage GraphicObjects. Updates three layers of GraphicObjects.

Static Objects: Usually stationary GraphicObjects without need for constant updates.

Dynamic Objects: Contains interactive GraphicObjects, updates frequently. ie. characters and interactive blocks.

CoverMapObjects: Top layer of GraphicObjects. Mostly for display and effects.

Handles the updates for GraphicObjects with the UpdateManager Class. Draws itself when given a canvas to be drawn on.

* + - * + Variables:

String title;

GraphicAnimation mapImage;

ObjPosition dimenstions;

ArrayList<GraphicObject> staticMapObjects;

ArrayList<GraphicObject> dynamicMapObjects;

ArrayList<GraphicObject> coverMapObjects;

UpdateManager updateManager;

Paint paint;

Boolean needsGraphicUpdate;

* + - * + Methods:

void addStaticObject(GraphicObject object): adds a graphicObjects to the staticMapObjects list

void addDynamicObject(GraphicObject object): adds a GraphicObject to the dynamicMapObjects list

void updateObjects(double inc): calls the update method in the updateManager. Updates all GraphicObjects in the Lists.

void draw(Canvas canvas): draws the panel and all the objects in the ArrayLists

void setMapPosition(double x, double y, AndroidConfig config): sets the x and y position for the panel and makes sure it stays in screen.

void setMapPosition(double x, double y): sets the x and y position for the panel

void setFrame(AndroidConfig config): calls updateManager.setFrame() based on the config data in AndroidConfig

void setFrame(int w, int h) calls updateManager.setFrame() based on gived dimensions.

int getHeight(): returns height

int getWidth(): returns width

int getxScreen(): returns x coordinate relative to the aspect ratio in AndroidConfig

int getyScreen(): returns y coordinate relative to the aspect ratio in AndroidConfig

int getHeightScreen(): returns height relative to the aspect ratio in AndroidConfig

int getWidthScreen(): returns width relative to the aspect ratio in AndroidConfig

* + - * **GraphicObject:**
        + Creates an Graphical Object to be manipulated by the custom SurfaceHolder, ViewPanel. Contains methods to set position, set velocity, and check collision. Comparator method to sort based on x position. Only updates when it is told to such as when the velocity is not 0 changes. Updates based on time passed.
        + Variables:

ObjPosition position;

int solidity; used to determine how to handle collision detection

GraphicAnimation image;

Paint paint;

int numOfCollisions; number of collisions with other GraphicObjects

ArrayList<ObjectStatus> listOfStatus; contains any statuses to be checked during update

boolean facingRight; determines direction object is facing

boolean needsGameUpdate; determines if update is needed

double xVel, yVel; x,y velocity to update by.

* + - * + Methods:

void setVelocity(double x, double y): sets velocity of object and sets needsGameUpdate to true

void setPosition(double x, double y): sets object position and sets needsGameUpdate to true

void addStatus(ObjectStatus status): adds object status to the listOfStatus

void update(double inc, ArrayList<GraphicObjects>[] objects): updates the GraphicObject based on all of the other GraphicObjects inside the Array of ArrayLists. Collision detection is detected here. Sets x and y position based on velocity. Reduces xVelocity each update until 0.

void checkStatus(): meant to be Overriden. called each update to check the listOfStatus

boolean checkCoordinateCollision(Coordinate coords): returns true or false based on if the given coordinates overlap the area inside the GraphicObject

boolean willTouch(GraphicObject object, double x, double y): returns true or false based on if the current GraphicObject will overlap object when at the coordinates x and y.

Coordinates checkCollision(GraphicObject object, Coordinates xy): called when willTouch() returns true during update. xy is the temporary coordinates that will be set to new coordinates at the end of update(double inc, ArrayList<GraphicObjects>[] objects). Can be Overriden to set the future xy coordinates based on collision.

void isTouching(GraphicObject object): called when object touches this GraphicObject. Meant to be Overriden

void draw(Canvas canvas): draws image.getFrame() onto screen or a solid rectangle if no image.

void drawRelativeToCoords(Canvas canvas, int x, int y): draws image.getFrame() relative to the given x, y coordinates.

return methods

* + - * + Class Comparator<GraphicObject> XValueComparator

Sorts based on X value of the left side

* + - * + Class Comparator<GraphicObject> XWidthValueComparator

Sorts based on X value of the right side

* + - * **ObjectStatus:**
        + Class to hold final values for statuses of GraphicObjects. Checked during updates. Meant to be implemented for custom statuses. Instantiated as objects.
        + Variables:

static final String IS\_ALIVE = “IS ALIVE”; indicates that the GraphicObject is currently alive

final String value;

* + - * + Methods

String getValue(): returns value of the ObjectStatus

* + - * **ObjPosition:**
        + Extends from Coordinates class. Contains width, height, collision ratios, and an AndroidConfig. Takes a AndroidConfig object to scale GraphicObjects by screen size. Scales height, width, x-coordinate, y-coordinate, for uniform look throughout devices.
        + Variables:

static AndroidConfig androidConfig; used to provide ratio for dimensions

double xScreen, yScreen; x, y value relative to aspect ratio

double height, width;

double heighScreen, widthScreen; height, width relative to aspect ratio;

double radius; space between center and the farthest point of the object

double xCollisionRatio; the x collision ratio defines the percentage from the left and right in which objects are solid

double yCollisionRatio; the y collision ratio defines the percentage from the top and bottom in which objects are solid

* + - * + Methods:

static void setAndroidConfig(AndroidConfig a): sets the AndroidConfig for the class

void setPosition(double x, double y): sets the x and y cords. also calculates the xScreen and yScreen coord

void setDimension(int w, int h): sets the width and height and updates the widthScreen and heightScreen

void setCollisionRatio(double xCol, double yCol): sets the CollisionRatio for x and y. defined between 0 and 0.5

Return Methods

* + - **AndroidConfig**:
      * Organizes screen size information to be used to scale objects given a resolution to be scaled to.
      * Variables
        + static AndroidConfig androidConfig; singleton AndroidConfig
        + DisplayMetrics displayMetrics; used to get display information from Activity
        + Activity activity;
        + int screenHeight; height of screen;
        + int screenWidth; width of screen;
        + double resolutionWidth: Scaled screenWidth
        + double resolutionHeight: scaled screenHeight
        + double aspectRatio: ratio of the screen from the resolution
      * Methods
        + public static void createAndroidConfig(Activity activity): creates and sets a new androidConfig.
        + static AndroidConfig getConfig(): returns singleton androidConfig
        + void setResolution(int wid, int hei): sets the width and the height of the given resolution
        + void updateAspectRatio(): updates the AspectRatio based on current resolution.
        + void printConfigData(): prints config data onto console;
        + Return Methods:
    - **ViewPanel:**
      * Abstract class extended from android.view.SurfaceView as a custom SurfaceView to handle GraphicObjects and GraphicBackgrounds. Contains a ThreadManager to organize the graphic thread and the update thread. UpdateGame() method(abstract) updates each tick from the update thread. UpdateGraphics() method updates all the graphics each tick.
      * Variables
        + ThreadManager threads;
        + SurfaceHolder surfaceHolder;
        + static Canvas canvas;
        + AndroidConfig androidConfig;
      * Methods
        + abstract void checkInput(); called every update to check userInput
        + abstract void updateGame(double updateIncrament): called every gameUpdate
        + void updateGraphics(double updateIncrament): called every graphicUpdate. calls the draw method in ViewPanel
        + void draw(Canvas canvas): draws the canvas
        + public void stopLoop(): stops all threads in the threadManager
        + void surfaceCreated(SurfaceHolder holder): called when the surface is created
        + abstract surfaceChanged(SurfaceHolder holder): called when the surface is changed
        + abstract void surfaceDestroyed(SurfaceHolder holder): called when the surface is destroyed.
  + Processes:
    - **GameLoop**:
      * Extends from Thread class to create a new thread for a game loop. Loop can run uncapped or with a set amount of ticks per second. If the loop has a define tick per second, sleeps based on the remaining time in each second. Each tick calls the check input method and the update method in the ThreadManager class.
      * Variables:
        + int maxFPS; defines maxFPS;
        + boolean powerSavingMode; defines if loop is uncapped
        + double averageMillisPerFrame; average millis per frame based on averagefps
        + int averageFPS; defines the average frames per second
        + boolean isRunning; defines if loop is running
        + long currentTime, previousTime; used to track delta time
        + double millisPerTick; millisecond per frame based on maxfps
        + int totalFrames; total ticks. resets every second
        + ThreadManager threadManager
      * Methods:
        + void run(): called on start() of new Thread;
        + void checkInput(): calls threadManager.checkInput()
        + void tick(double updateIncrament): calls threadManager.updateGame(updateIncrament)
        + togglePowerSavingMode(): switches powersavingmode between true and false
        + void stopGameLoop(): stops loop
        + double toMillis(long nanotime); converts nanotime to millis
        + double toAverageMillisPerFrame(double time, int totaleFrameS): calculates averageMillisPerFrame based on time and totalframes
    - **GraphicsLoop:**
      * Identical to the GameLoop but updates the graphic update method in the ThreadManager class. Does not include CheckInput method.
    - **ThreadManager:**
      * Contains a GameLoop thread and a GraphicsLoop thread to handle tick updates. Is used by the ViewPanel to organize the updates for all of the GraphicObjects inside the ViewPanel.
      * Variables:
        + GameLoop gameProcess;
        + GraphicsLoop graphicProcess;
        + ViewPanel panel;
      * Methods
        + void start(); starts both gameProcess and graphicProcess
        + void checkInput(): calls panel.checkInput(). called each update in the gameProcess
        + void updateGame(double inc): calls panel.updateGame(inc). called each update in gameProcess
        + void updateGraphics(double inc: calls panel.updateGraphics(in). called each upate in graphicProcess
        + void stopThreads(): stops both gameProcess and graphicProcess
        + Return Methods
    - **UpdateManager:**
      * Organizes all of the GraphicObjects in a given list of lists and updates them based on the updates of GraphicBackground. Is used to sort the position of all of the objects and compare collision based on position.
      * Variables
        + ArrayList<GraphicObjects>[] objects;
        + ArrayList<GraphicObject> visibleObjects;
        + ObjPosition frame;
      * Methods
        + setFrame(ObjPostion f): sets the frame in which objects should be visible
        + updateFrame(double x, double y): relocates frame of visibility
        + sortLists(): sorts all of the GraphicObjects in objects based on XWidthComparitor in GraphicObjects
        + SortList(ArrayList<GraphicObject> list): sorts given list based on XWidthComparitor
        + void printList(ArrayList<GraphicObject> list): prints out list
        + void printLists(): prints out all lists in objects
        + void updateGame(double inc): updates all objects in the list when called by increment of time
* Oblivion:
  + These set of classes are used to demonstrate the usage of the classes in androidEngine.
  + Activities
    - **GameActivity:**
      * Extends from android.app.Activity to start an Activity containing a GamePanel. On Creation, creates a GamePanel and starts the threads for GamePanel.
    - **LoadingActivity:**
      * Extends from android.app,Activity to start an Activity for loading in assets for the game. Creates and populates data for AndroidConfig with the physical screen size.
    - **MenuActivity (inactive):**
      * Currently empty
  + Mainframe
    - Game Components
      * Graphic Components:
        + **BlockStatus:**

Extends from ObjectStatus to define final variables for ObjectStatuses for custom GraphicObjects.

* + - * + **Button (inactive):**

Extends from GraphicObject to create an interactive button to be put onto a ViewPanel. Currently Empty.

* + - * + **CharacterObject:**

Extends from GraphicObject to create a moving character. Contains methods to control the character based on an arraylist of PointerGestures given. Will always update each tick.

* + - * + **SolidObject:**

Extends from GraphicObject to create a static solid block. Other GraphicObjects will not be able to move through the block unless specifically set got through the block.

* + - * **Equip (inactive):**
        + Currently empty
      * **GameCharacter (inactive):**
        + Currently empty
      * **Item (inactive):**
        + Currently empty
    - Game Data
      * **FileDataManager (inactive):**
        + Contains a FileReader and FileWriter object to read and write cache files. Reads and writes multiple different files when given a file name. Writes based on an array of strings. returns an ArrayList of strings when asked to read the file.
      * **GameData:**
        + Contains all of the data for GraphicObjects and GraphicBackgrounds to be created in the game. Also contains the pointers to Image resources.
      * **GameManager:**
        + Takes data from GameData and returns Objects and Backgrounds created from the data.
    - Graphics
      * **GameAction:**
        + Set of Enums to define different actions for a character to read.
      * **GamePanel:**
        + Extended from ViewPanel. Creates a CharacterObject and adds it to the GraphicBackground. Handles touch events and defines PointerGestures for the CharacterObject to read.
      * **MenuPanel:**
        + Currently Empty.
      * **PointerGesture:**
        + Organizes data for individual pointers on the screen. Contains a single Game Action based on pointer coordinates. Each individual pointer is defined with an ID given to them by the MotionEvent.