COINCRAVER DEVELOPMENT GUIDE

CSC 780 under Dr. Puder, San Francisco State University

Things that we will learn:

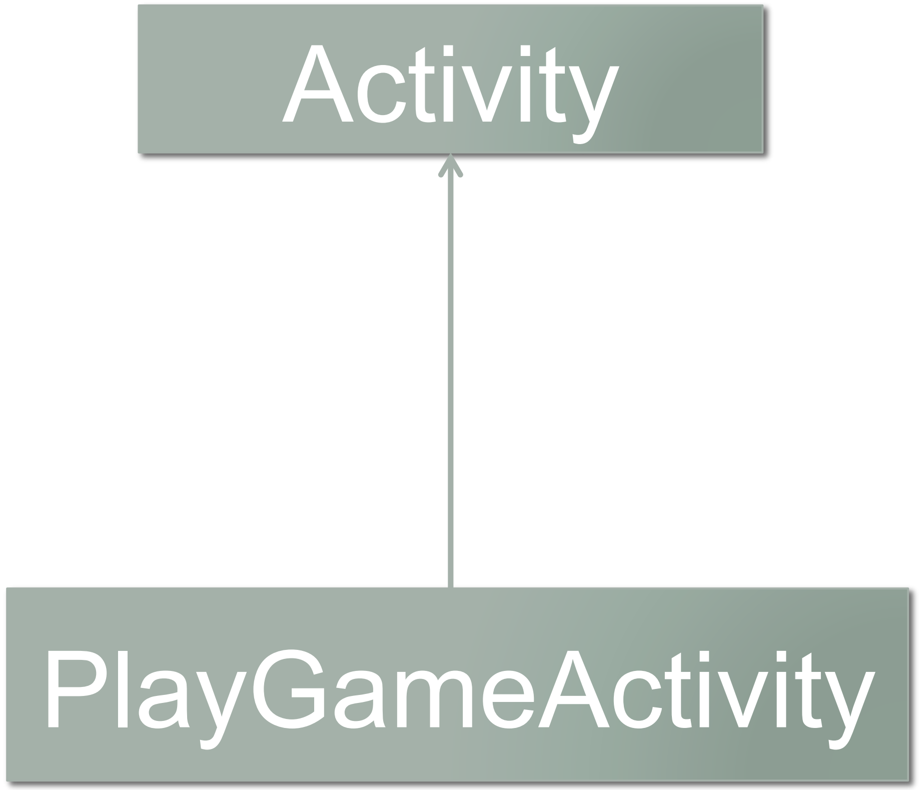
- How to setup the Coincraver Game Loop?
- How to draw using the canvas?
- How to take the timings ? (many ways)
- How to handle collisions between the objects in Coincraver?
- How to have fun ? ☺

Android & Game Classes in Coincraver

- Android classes:
  - *PlayGameActivity*
  - *GameViewActivity*
  - *Class PlayerMoveThread (extends thread)*
    - *Defined in GameViewActivity*

- Game classes:
  - *GameActivity*
  - *PlayerActivity*

Create “PlayGameActivity” of Coincraver

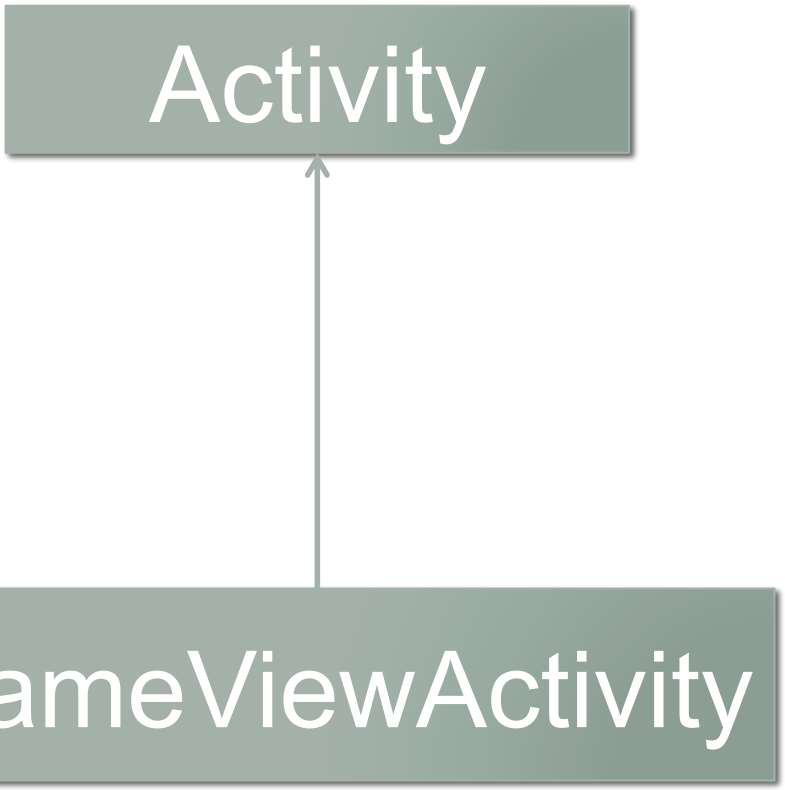


Now we have a activity with something like Hello World!…..

Do we need a View for Coincraver?

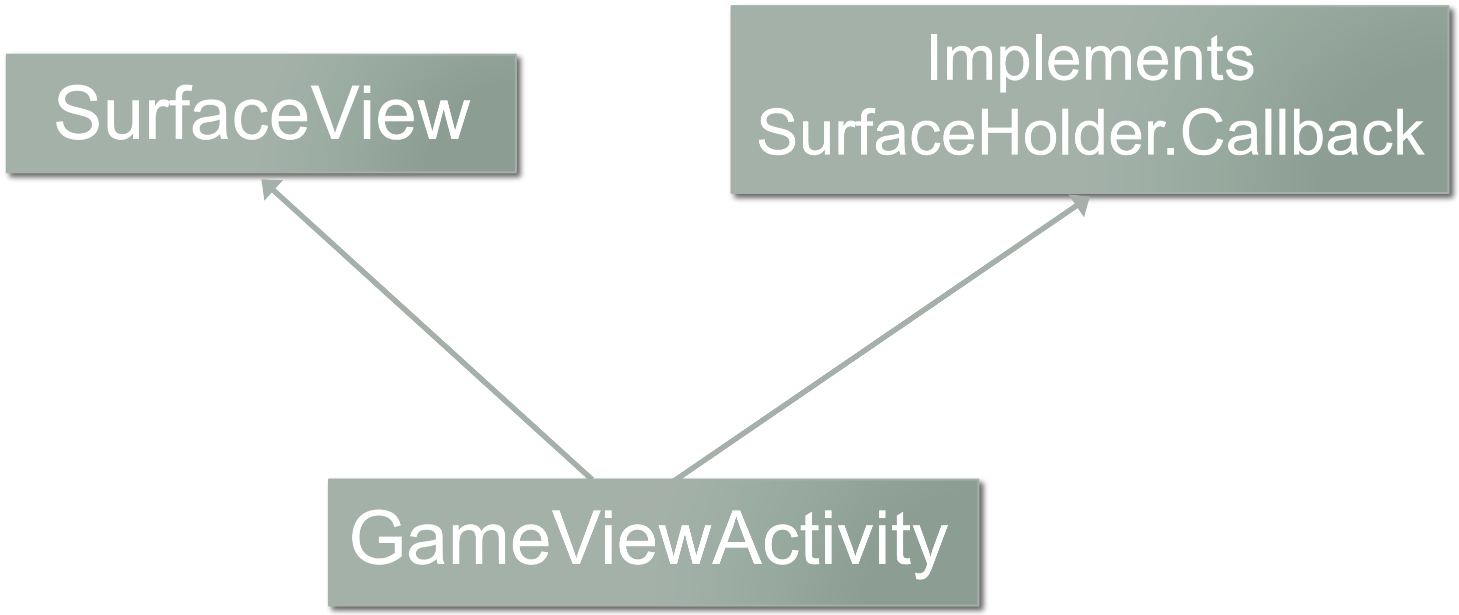
- Yes, we need a view so that we can house our Coincraver
- Specifically, what we need is a View which allows us to continuously draw to a screen and get player input
- Do we have an option? Yes. There is a SurfaceView class for this purpose (there is a nice Camera tutorial written by Dr. Puder that uses SurfaceView class)
- SurfaceView class is provided by Android SDK

Create “GameViewActivity” of Coincraver



Now, we have a view that can extend SurfaceView and implement callbacks

Structure will look like this…



Callbacks are important:

1. To know when the surface is created so that we can start our Coincraver (do you remember SurfaceCreated( ) callback method in Camera tutorial of Dr. Puder, check that out if not)
2. To know when the surface is being destroyed so that we close our Coincraver.
3. To know when the surface size changes so we can get the updated display dimensions.

Why do we need to extend SurfaceView?

- It provides the <u>drawing surface</u> (like a drawing board)
- This surface, then, can be modified with the help of Canvas class
- It also allows <u>any other thread to render to the view</u>
- This is required if we want Coincraver to run independent of the view UI (that is we can have animation in the background later on in the future)

PlayerMoveThread in GameViewActivity

- It will be responsible for interfacing our Coincraver game code with Android
- Also, we will have to make sure that GameViewActivity is our default view of Coincraver (..after the Start Game button is clicked)

Let’s focus on the game code now..

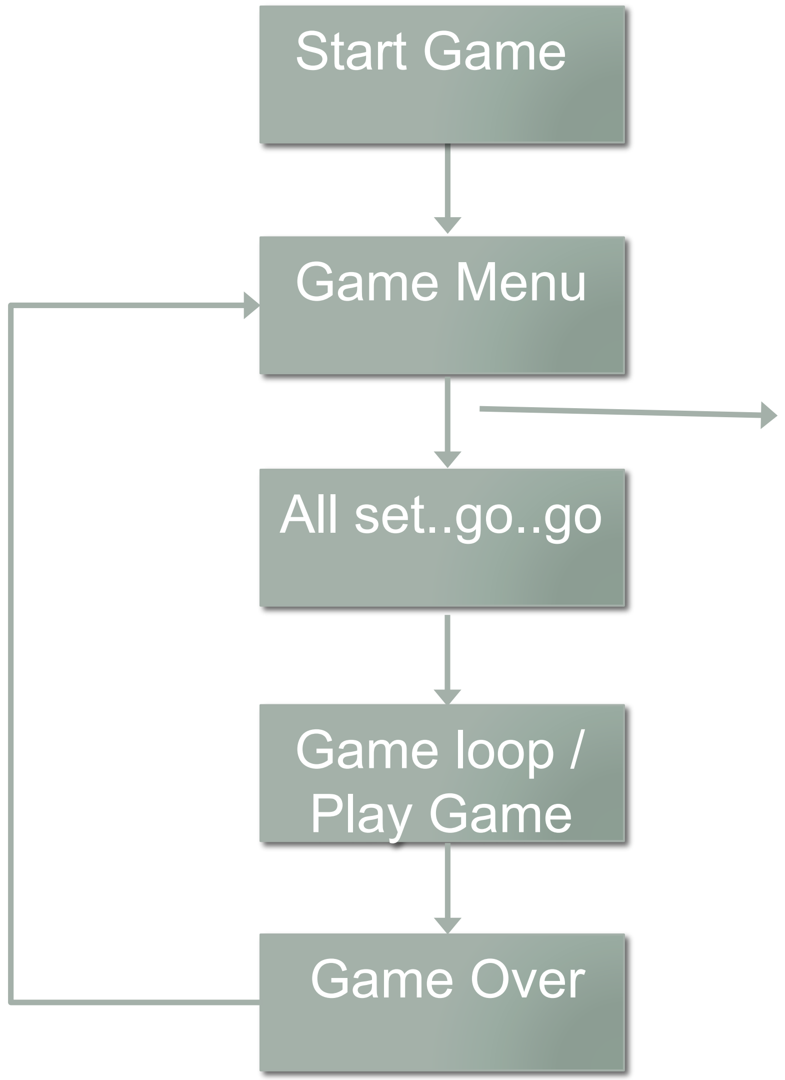
GameActivity → It is responsible for

1. Managing the game resources
2. Keeps track of the game state (used switch case)
3. Handles the game logic

PlayerActivity → It is responsible for

1. Drawing the player
2. Updates the player logic
3. <u>(collision detection, not yet implemented</u>)

Which states do we need to manage?



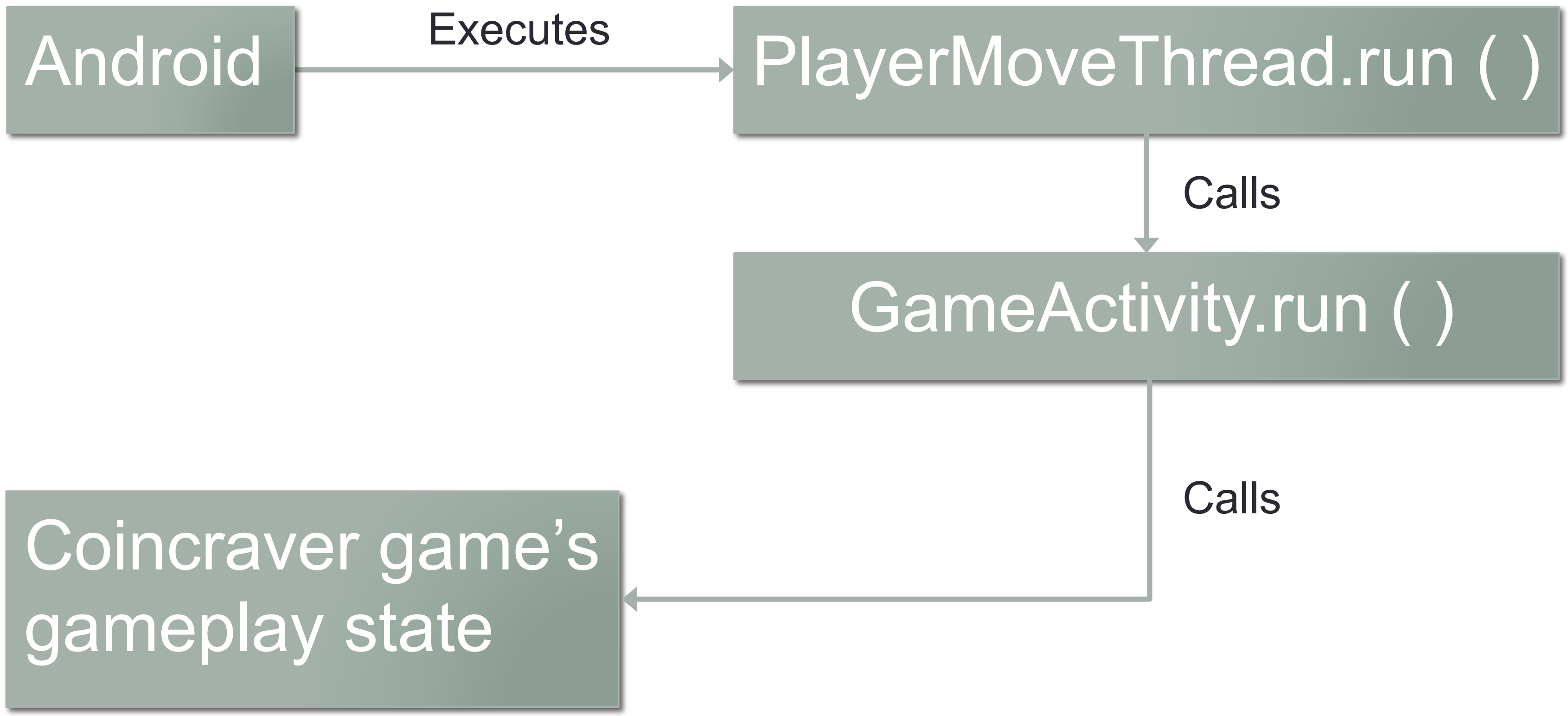
Player has pressed the "Start Game" button

PlayerActivity will look like this..

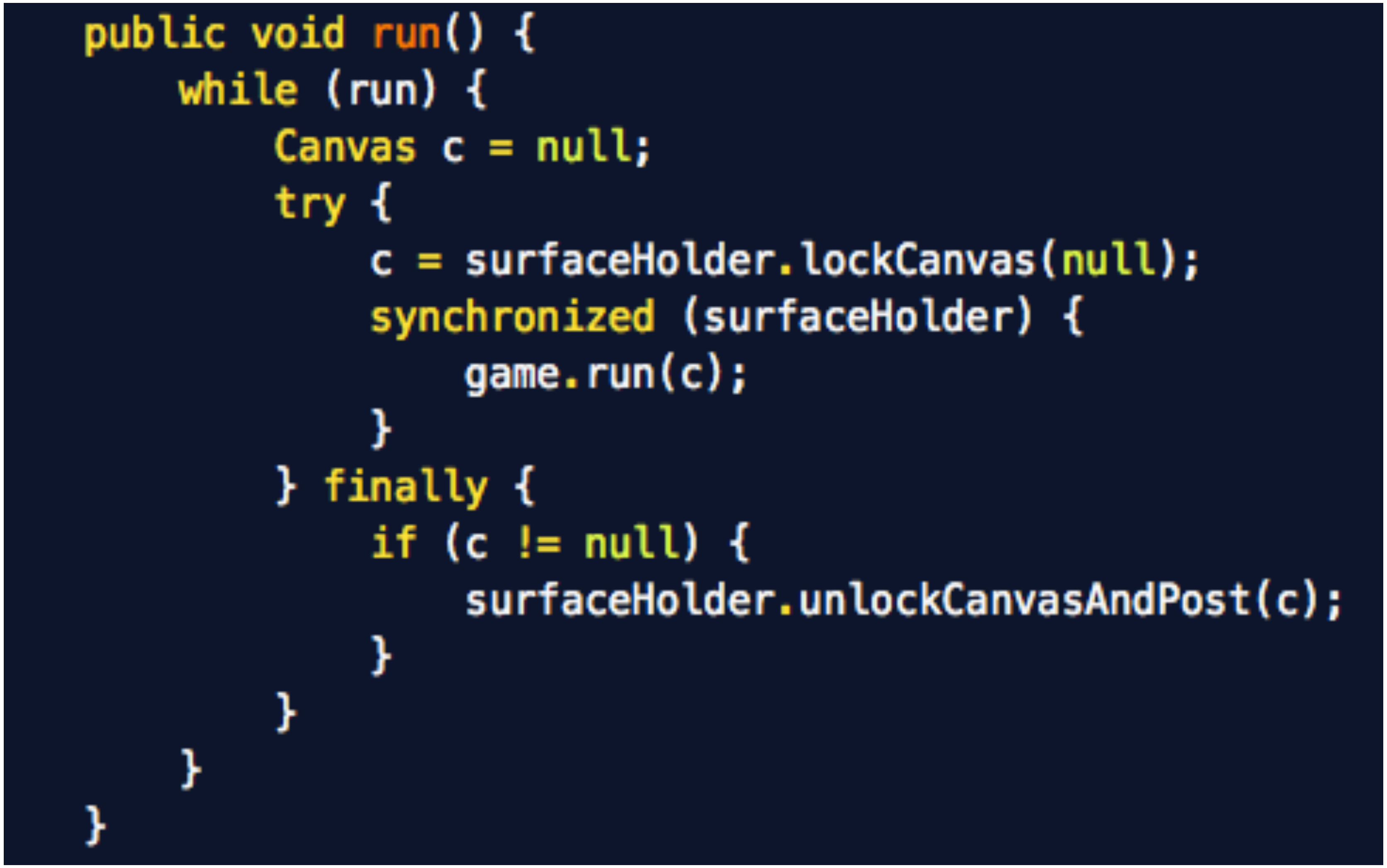
```
class PlayerActivity
{
        public PlayerActivity(GameActivity game)
        {
                }
        public void reset()
        {
                }
        public void update()
         {
                }
        public void draw(Canvas canvas)
         {
                }
}
```

Setting up the Coincraver game loop..

- PlayerMoveThread will play a major role.

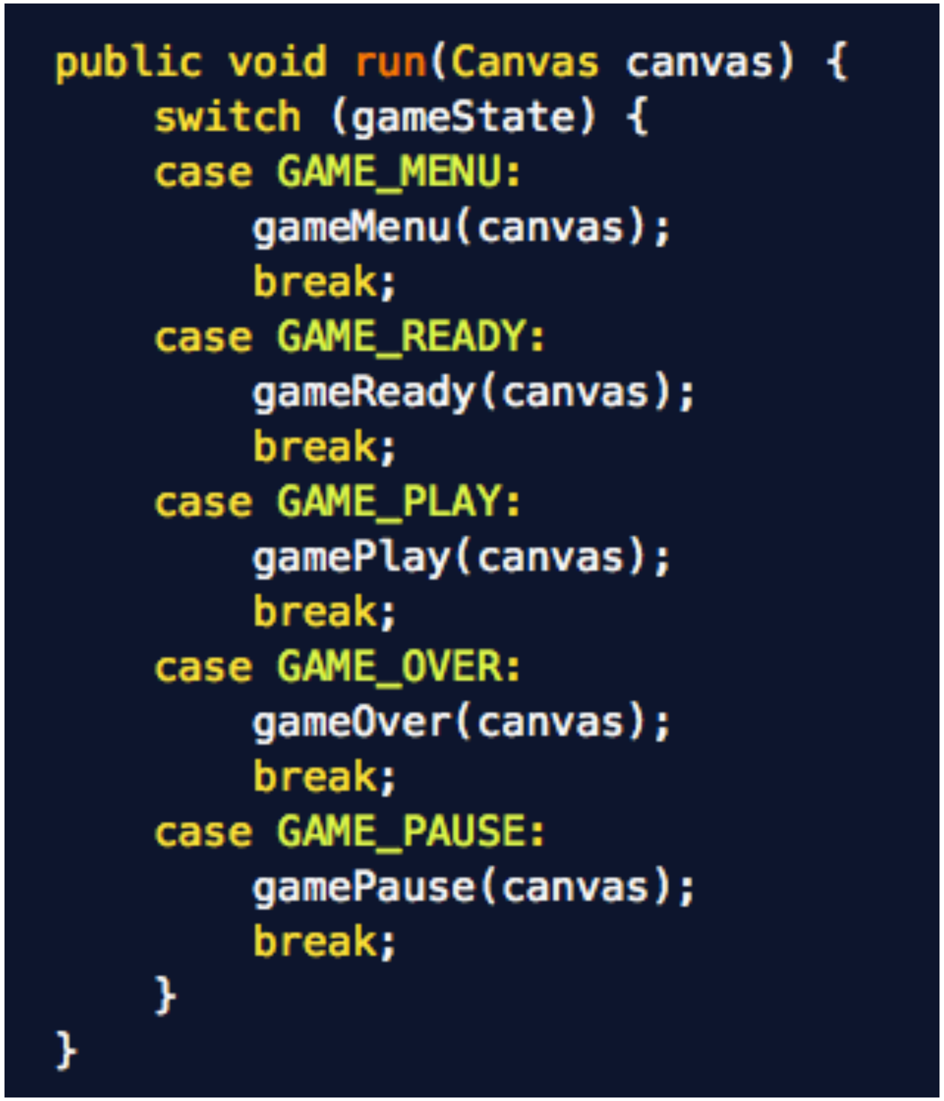


PlayerMoveThread.run ( ) method



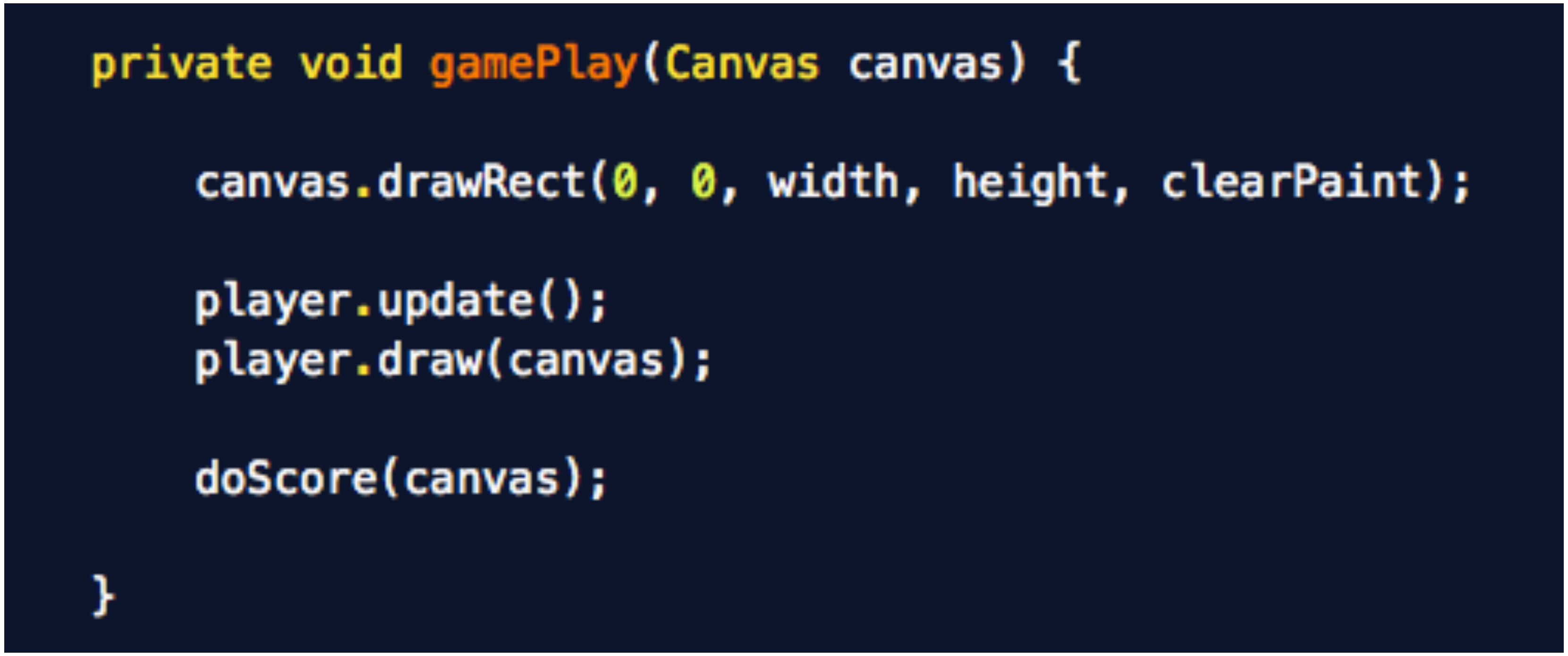
```
public void run() {
    while (run) {
        Canvas c = null;
        try {
            c = surfaceHolder.lockCanvas(null);
            synchronized (surfaceHolder) {
                game.run(c);
            }
        } finally {
            if (c != null) {
                surfaceHolder.unlockCanvasAndPost(c);
            }
        }
    }
}
```

GameActivity.run ( ) method



```
public void run(Canvas canvas) {
    switch (gameState) {
    case GAME_MENU:
        gameMenu(canvas);
        break;
    case GAME_READY:
        gameReady(canvas);
        break;
    case GAME_PLAY:
        gamePlay(canvas);
        break;
    case GAME_OVER:
        gameOver(canvas);
        break;
    case GAME_PAUSE:
        gamePause(canvas);
        break;
    }
}
```

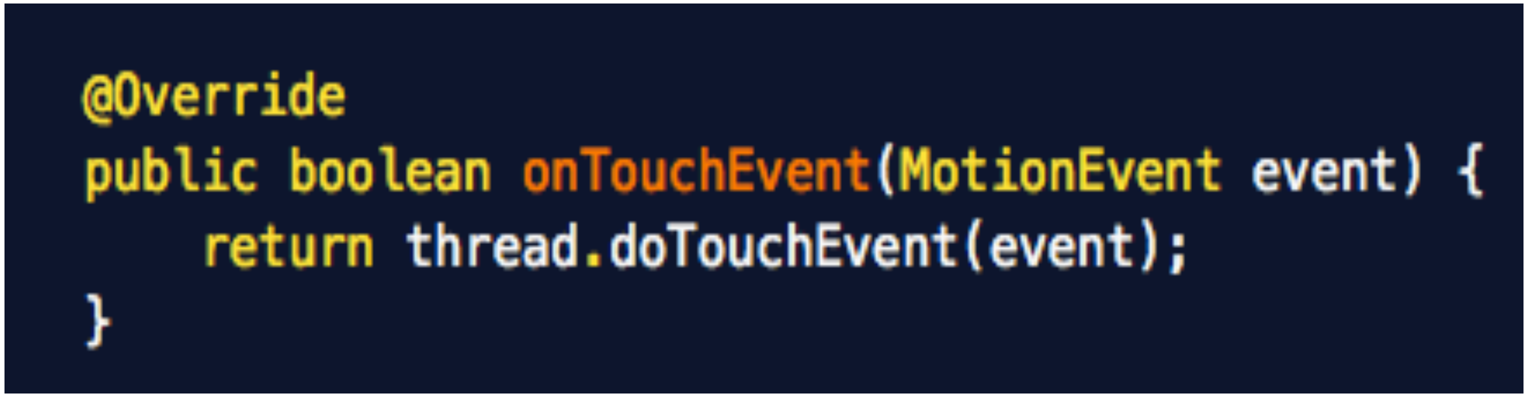
GameActivity’s gameplay ( ) method



```
private void gamePlay(Canvas canvas) {

    canvas.drawRect(0, 0, width, height, clearPaint);

    player.update();
    player.draw(canvas);

    doScore(canvas);

}
```

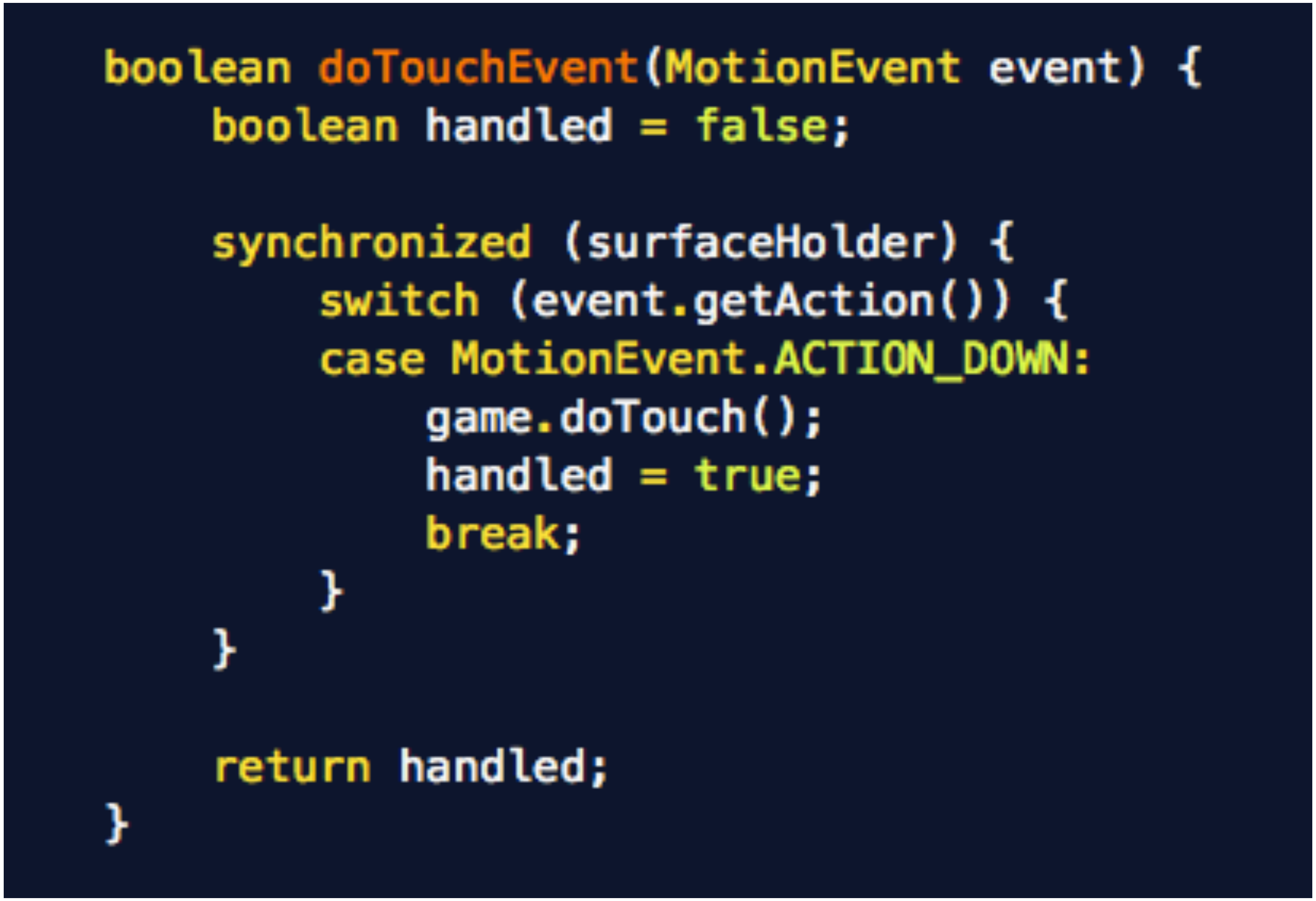
Where is the input ?

- Input comes from the UI thread from the default view
- GameViewActivity handles touch events and notifies the game when a touch occurs
- GameViewActivity’s touch event handler:



```
@Override
public boolean onTouchEvent(MotionEvent event) {
    return thread.doTouchEvent(event);
}
```

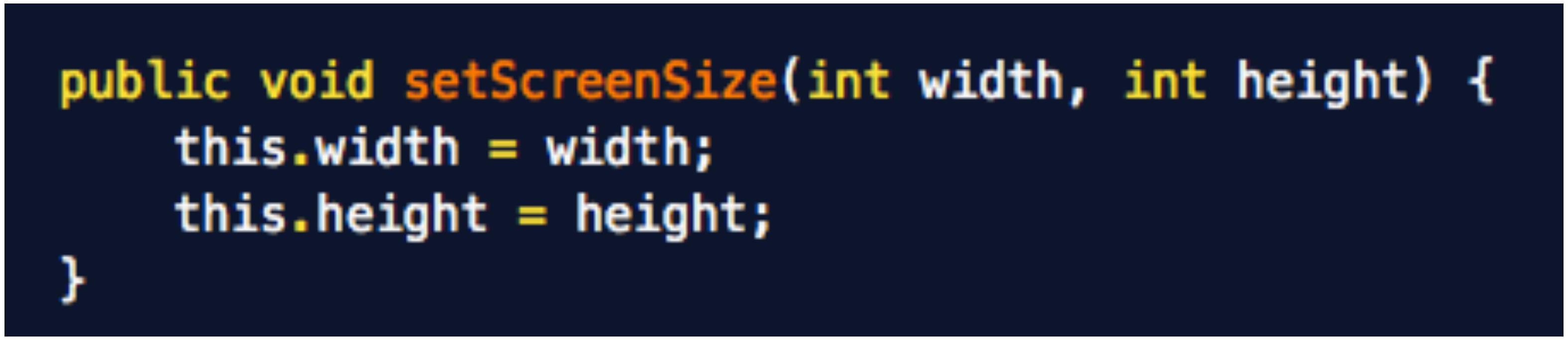
PlayerMoveThread’s touch event handler



```
boolean doTouchEvent(MotionEvent event) {
    boolean handled = false;

    synchronized (surfaceHolder) {
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            game.doTouch();
            handled = true;
            break;
        }
    }

    return handled;
}
```

Building of the GameActivity class

- Now, try to run Coincraver game loop. So, draw something and use timing
- To draw something, we need the screen size. How to get it?
  - SurfaceView callbacks notify our view of the screen's size. We need to pass this information in the GameActivity class at the runtime. Therefore, setting up the setScreenSize( ) method in GameActivity.java:



```java
public void setScreenSize(int width, int height) {
    this.width = width;
    this.height = height;
}
```