GAME DEVELOPMENT FRAMEWORK

Ho Dac Hung

Purpose

- There are many tasks that every game must be able to perform.
- The role of a game development framework is to provide reusable classes that perform such tasks, so that you, as a game programmer, can focus on writing the game-specific code that will make or break your game.

Good game development framework

- There is no correct way to create a game development framework.
- A good game development framework should be flexible.

Game loop

- The game loop is the heartbeat of every game. We used a very rudimentary one so without any control over how fast or slow we update our game state and which frames to render.
- FPS – Frames per Second
- UPS – Update per Second

Designing our framework

- Main classes
- State classes
- Utility classes
- Animation classes

Main classes

- GameMainActivity
- GameView
- Assets

State classes

- State
- LoadState
- MenuState

Utility classes

- InputHandler
- RandomNumbergenerator
- Painter

Animation classes

- Animation
- Frame

Create GameMainActitivy

- Extend Activity and override an onCreate() method.
- In our Manifest, we set our new GameMainActivity as the launcher Activity, so that it becomes the starting point of our application.
- We set android:screenOrientation as “sensorLandscape”.
- We use the android:theme attribute to remove the title bar and set our application to full screen using the built in style @android:style/Theme.NoTitleBar.Fullscreen.

Create GameView

- Create a new class called GameView which extends SurfaceView.
- We will make use of the gameWidth and gameHeight values.

Setting the GameView as the Content View

- Start by navigating to the GameMainActivity and declaring the following class variables, importing android.content.res.AssetManager.

```
public static final int GAME_WIDTH = 800;
public static final int GAME_HEIGHT = 450;
public static GameView sGame;
public static AssetManager assets;
```

Setting the GameView as the Content View

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        assets = getAssets();
        sGame = new GameView(this, GAME_WIDTH,
GAME_HEIGHT);
        setContentView(sGame);
}
```

GameView's components

- Current state
- Handling input
- Handling drawing
- Canvas and Memory management
- Screen resolution vs. Game resolution

Painter class

- The purpose of this class is to make the rendering process in our Android framework.

```
private Canvas canvas;
private Paint paint;
private Rect srcRect;
private Rect dstRect;
private RectF dstRectF;
```

State class

- setCurrentState
- init
- update
- render
- onTouch

InputHandler class

- setCurrentState
- onTouch

Assets class

- SoundPool
- Bitmap
- load
- loadBitmap
- loadSound
- playSound

State classes

- LoadState
- MenuState
- PlayState
- GameOverState
- …

GameView class

```
private Bitmap gameImage;
private Rect gameImageSrc;
private Rect gameImageDst;
private Canvas gameCanvas;
private Painter graphics;
private Thread gameThread;
private volatile boolean running = false;
private volatile State currentState;
private InputHandler inputHandler;
```

GameView class

```
gameImage = Bitmap.createBitmap(…);
gameImageSrc = new Rect(…);
gameImageDst = new Rect();
gameCanvas = new Canvas(gameImage);
graphics = new Painter(gameCanvas);
```

GameView class

- When working with a surface such as SurfaceView, we must be careful not to start rendering too early and stop rendering too late. An Android application switches from one Activity to another, meaning that our SurfaceView may be created and destroyed at our player's whim.

Setting up Input

```
private void initInput() {
    if (inputHandler == null) {
        inputHandler = new InputHandler();
    }
    setOnTouchListener(inputHandler);
}
```

Setting the Initial State

```
public void setCurrentState(State newState) {
    System.gc();
    newState.init();
    currentState = newState;
    inputHandler.setCurrentState(currentState);
}
```

Implementing game loop

- Implement the Runnable interface.
- Add the unimplemented run() method.
- Create the methods initGame() and pauseGame().
- Call the initGame() and pauseGame() methods in surfaceCreated() and surfaceDestroyed().
- Create methods updateAndRender(long delta) and renderGameImage().
- Implementing run() method.

Frame, Animation

- Frame holds bitmap and duration.
- Animation holds frames and render based on duration.

RandomNumberGenerator

- RandomNumberGenerator generates random number.