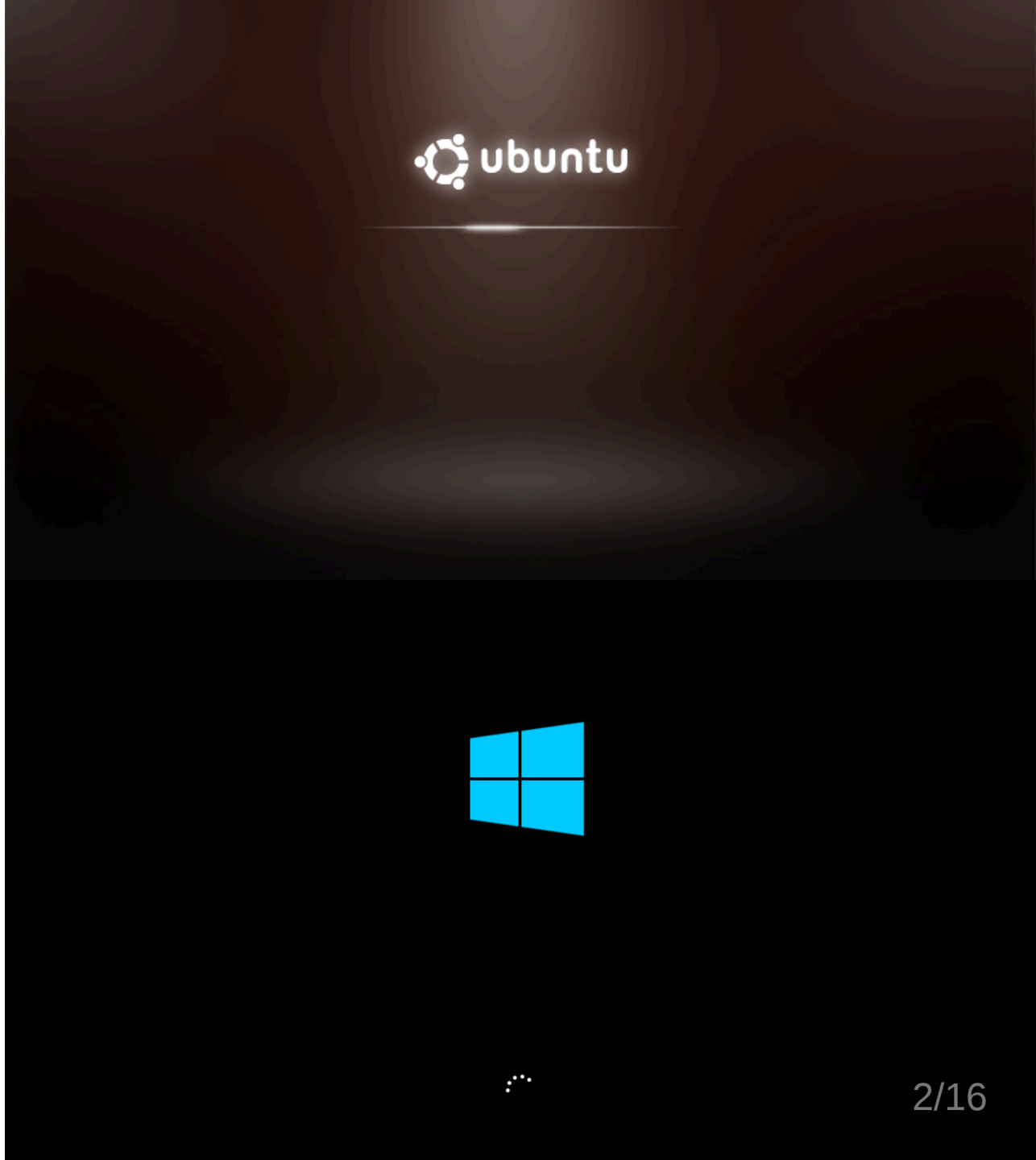
Splash Screens

```
Module Code: ELEE1146

Module Name: Mobile Applications for Engineers

Credits: 15

Module Leader: Seb Blair BEng(H) PGCAP MIET MIEEE MIHEEM FHEA
```

Download as a PDF

Bootsplash

- AKA bootscreen
- Graphical representation Boot process of the OS
- is not necessarily designed for marketing purposes;
- it can be intended to enhance the experience of the user as eye candy
- or provide the user with messages (with the added advantage of color-coding facility) to diagnose the state of the system.

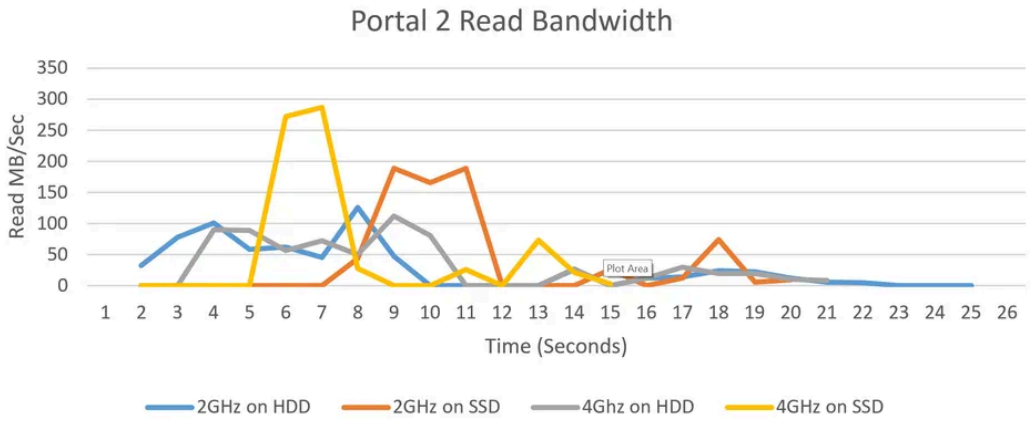


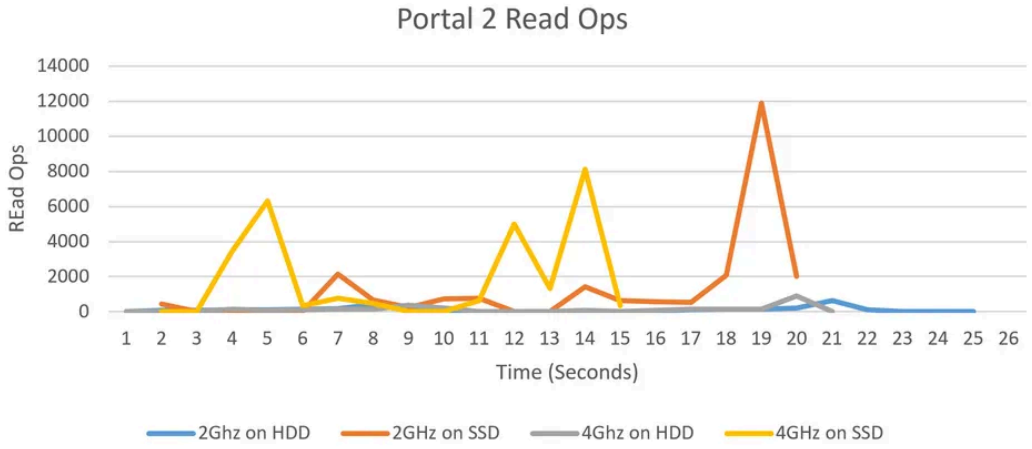
Interstitial Webpage

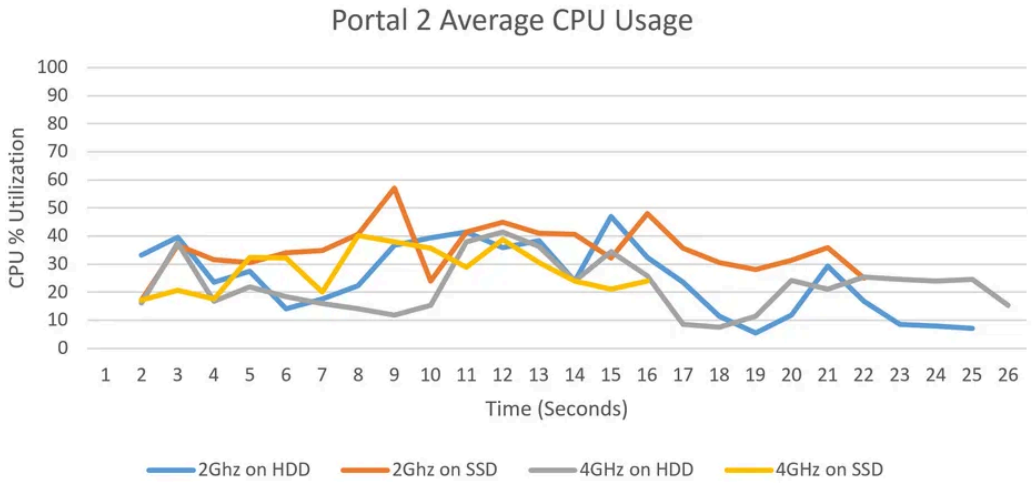
- Interstitial (interval or intervening space or segment)
- displayed before or after the expected content page

Loading Screen

- is a screen shown by a computer program, very often a video game, while the program is loading (moving program data from the disk to RAM) or initialising.
- Some loading screens display a progress bar or a timer countdown to show how much data has actually loaded. Others, recently, are not even a picture at all, and are a small video or have parts animated in real-time.







Loading Real World: Portal 2

Android Splash Screen

```
private val SPLASH_DELAY: Long = 4000
Looper.myLooper()?.let { looper ->
    val handler = Handler(looper)
    handler.postDelayed({
      // Create an Intent to launch the MainActivity
      val intent = Intent(this, MainActivity::class.java)
      startActivity(intent)
      finish() // Finish the splash screen activity
    }, SPLASH_DELAY)
} // Number is in milliseconds.
```

`Handler` Class

Two main instances...

1. To schedule messages and runnables to be executed at some point in the future
2. To enqueue an action to be performed on a different thread than your own.

Send and notify state changes of your objects to other applications using an Event-driven Architecture

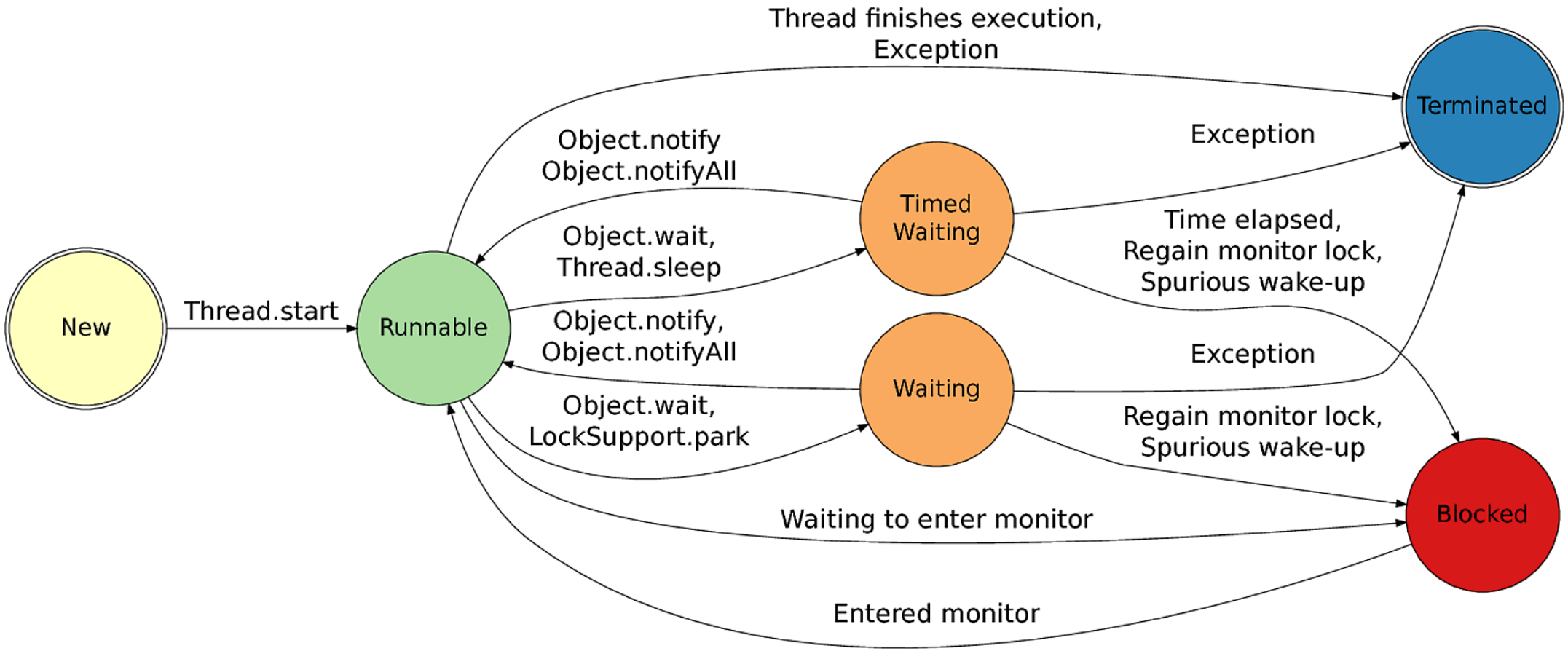
- you want to create a loosely coupled system
- you want to build a more responsive system
- you want a system that is easier to extend

> ref -> `https://developer.android.com/reference/android/os/Handler.html`

`Handler` Class Diagram


center

Threads/Runnerable



Animations - Transitions Functions

- `overridePendingTransition(enterAnim:Int, exitAnim:Int)` - API 33 and below
- `overrideActivityTransition(overrideType:Int, enterAnim:Int, exitAnim:Int)` - API 34
  - `enterAnim`: A resource ID of the animation resource to use for the incoming activity. Use 0 for no animation.
  - `exitAnim`: A resource ID of the animation resource to use for the outgoing activity. Use 0 for no animation.
  - `OVERRIDE_TRANSITION_OPEN/CLOSE`: starting/entering or finishing/closing

Both functions are used to customise the animation for the activity transition with this activity. This can be called at any time while the activity is still alive.

Using `overridePendingTransition()`

- Store all animation files in the `/res/anim` directory, you will need to create this yourself.

```
overridePendingTransition(0,R.anim.slide_in_left)

// fade_out/fade_in are built-in animations
overridePendingTransition(android.R.anim.fade_out,android.R.anim.fade_in)
```

Animations Types:

- Slide -> Vertical(up, down) and Horizontal(left, right)
- Zoom/Scale
- Rotation -> Including spiraling
- More here:
  - https://developer.android.com/guide/topics/resources/animation-resource

Animation configuration XML (Sliding)

- `<translate>` for sliding
- `fromXDelta` Change in X coordinate to apply at the start of the animation
- `toXDelta` Change in X coordinate to apply at the end of the animation
- `fromYDelta` Change in Y coordinate to apply at the start of the animation
- `toYDelta` Change in Y coordinate to apply at the end of the animation

```
<!--Slide out Left-->
<translate
    android:fromXDelta="100%"
    android:toXDelta="0"
    android:duration="3000" />

<!--Slide in Left-->
<translate
    android:fromXDelta="0"
    android:toXDelta="100%"
    android:duration="3000" />
```

Scaling

- `fromX/YScale` - starting from N size offset, where 1.0 is no change
- `toX/YScale` - ending from N size offset, where 1.0 is no change
- `pivotX/Y` - coordinate to remain fixed when the object is scaled

```
<scale
  android:fromXScale="1.0"
  android:fromYScale="1.0"
  android:toXScale="0.0"
  android:toYScale="0.0"
  android:pivotX="50%"
  android:pivotY="50%"
  android:duration="1000" />
```

Scale in

Rotate

- `fromDegrees` - Starting angular position, in degrees
- `toDegrees` - Ending angular position, in degrees.

```
<rotate
  android:fromDegrees="0"
  android:toDegrees="359"
  android:pivotX="50%"
  android:pivotY="50%"
  android:duration="1000" />
```

Advanced animation: Interpolators

- An interpolator is an animation modifier defined in XML that affects the rate of change in an animation.
- This lets your existing animation effects be accelerated, decelerated, repeated, bounced, etc

Kotlin	XML
`AccelerateDecelerateInterpolator`	`@android:anim/accelerate_decelerate_interpolator`
`AccelerateInterpolator`	`@android:anim/accelerate_interpolator`
`AnticipateInterpolator`	`@android:anim/anticipate_interpolator`
`AnticipateOvershootInterpolator`	`@android:anim/anticipate_overshoot_interpolator`
`BounceInterpolator`	`@android:anim/bounce_interpolator`
`CycleInterpolator`	`@android:anim/cycle_interpolator`
`DecelerateInterpolator`	`@android:anim/decelerate_interpolator`
`LinearInterpolator`	`@android:anim/linear_interpolator`
`OvershootInterpolator`	`@android:anim/overshoot_interpolator`