SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Computing and Information Systems

[CS205] Operating Systems Concepts with Android

Assignment #4

Purpose of this exercise

Upon successful completion of this exercise students should be able to do the following:

1. Demonstrate familiarity with programming in Android environment.
2. Use Android's built-in and explicitly implemented mechanisms of multithreading and synchronization to enable non-blocking interactive execution.
3. Demonstrate knowledge gained through the course by delivering a working implementation of a practical application.

Overview

Although not necessarily mainstream 10-15 years ago, software development for mobile devices in now not just commonplace, but portable touchscreen computers have become the primary, and often the only target platform for many applications. In this space, there are two leading operating systems: Android and iOS, with their worldwide market share of nearly 71.7% and 27.8%, respectively (data as of Q1 2025). Ability to create software aimed at either of these OSes, familiarity with their features and ecosystem, is an important skill for computer scientists and software engineers.

Due to its dominance, in this assignment we will focus on Android. Android Studio is the official integrated development environment for writing applications targeting this platform, and so this IDE is a required component to complete this assignment. Android Studio can be used to effortlessly create programs using the rapid application development (RAD) methodology that employs drag-and-drop approach to building graphical user interfaces (GUI) with visual components. It can be also used to create complex experiences that generate their own 2D and 3D graphics. Through this exercise you will develop mostly the latter to better understand how multithreading and synchronization primitives enable fast, responsive and execution. The specific class of software that you will create is referred to as *real-time interactive simulations*. It includes entertainment software, games, as well as professional training simulators, and applications for modelling of real-world industrial processes and environments.

Through this assignment, you will get a chance to explore software programming in Android, and to put to use insights about processes and threads to create a working application: a real-time interactive simulation.

We define the principles expressed by the term “*real-time interactive simulation*” as follows:

- **Real-time** – events in a program happen following a certain continuous timeline. This timeline must be simulated with *discrete-time* processing steps performed for time deltas of fixed or variable rate. To ensure appropriate realism, the program must execute fast enough to simulate this continuous timeline at a smooth, constant pace, often the same as the natural time flow. Skilful use of multithreading or inter-process communication allows for accessing more computational resources and executing long-running algorithms in parallel.
- **Interactive** – events in a program develop regardless of whether the user of the program provides any input, or not. These interactions must be captured in a non-blocking fashion to ensure continuity of simulation. Organization of code to a structure where state updates (potentially costly) and synthesis of output (frequent) can be performed synchronously, but at a different pace, may be crucial.
- **Simulation** - events progress from the predefined *initial state*, and follow *deterministic rules* of interactions of objects being simulated, while the external environment (especially: human users) influence the state of a simulated world by triggering spontaneous interactions from external inputs.

Note that most operating systems, including iOS and Android, are not technically classified as *real-time operating system* (RTOS), as they do not offer strict guarantees about timing of execution, delays, and responsiveness. Their approach can be only described as “*best effort*” (they are more than sufficient to deliver non-critical simulations and entertainment, but definitely should not be used in automation of industrial or other highly sensitive processes).

Requirements

This is a group assignment. With your team you must develop a working Android application on a selected topic that meets the stipulated technical requirements. You can implement it in Java or Kotlin programming languages. A sample program in Java is presented in the laboratory exercise 3. For this assignment your team should target Android API level 34 (Android 14; 2023). Do not forget to configure your virtual simulator device to match this API.

This project aims to provide an engaging way for students to grasp fundamental concepts of operating systems while developing problem-solving, time-management, and strategic thinking skills, and giving students an opportunity to translate their insights into a fun and interactive Android application.

The theme for this term's submissions is:

> **The player is the operating system.**

In this year's assignment, the theme revolves around immersing players in the role of an operating system, where they must tackle the complexities of resource management. The game must challenge players to efficiently manage some limited computer resources (e.g. allocate CPU time, handle processes, copy memory, swap memory pages, juggle files, and ensure smooth system performance under pressure and constraints of time).

Looking for an inspiration? Perhaps players will face scenarios such as dealing with system overloads, resolving deadlocks, and optimizing task scheduling. As they progress through levels, the simulation might introduce increasingly difficult tasks, forcing players to think critically and balance competing demands to keep the system running smoothly.

Your team still does not know where to start? Try to warm up by playing this game: https://plbrault.itch.io/youre-the-os. Got inspired? Now it is your turn to create something original!

You have freedom to interpret this theme, and select a topic for your simulation. However, the application must be suitable for children, and may not include gore, violence, or other controversial elements.

The graded requirements are focused around the technical side of this assignment. Your team must demonstrate all 5 required features, and any 5 additional features as selected by your team and relevant to your application.

Required features

Fulfil **all** of the following requirements:

1. The application contains at least 1 activity using 2D graphics drawn by code.
2. The application works in “real-time”, and as such it includes 3 types of dynamic elements, progressing without human interaction:
    - Real-time elements updated synchronously in each frame and animated smoothly, fluently at a constant rate despite variable framerate, resulting from irregular *time deltas* between frames.
    - Interval elements updated synchronously at predetermined interval steps with each update spanning a fixed *time delta*.
    - Asynchronous elements updated in worker threads, regardless of a simulated *time delta*.
3. The application is “interactive”, and as such it includes behaviours triggered by user events (e.g. click, touch, key press, voice, captured image, etc).
4. The application performs parallel operations by creating at least one worker thread.
5. The application ensures that threads synchronize updates to a common state with built-in synchronization primitives, e.g. mutexes, monitors.

Additional features

Fulfil **5** distinctive of the following requirements:

6. The application contains at least 1 activity built with standard GUI components
(e.g. the *main menu* or the *authors*/*credits* activity).
7. The application uses a thread pool for performing tasks triggered by human interactions.
8. The application uses a *producer-consumer pattern* for performing asynchronous tasks.
9. The application utilizes in advanced ways built-in abstractions offered to facilitate multi-threading
(e.g. Looper class, message queues, asynchronous tasks, futures, background services).
10. The application uses non-trivial navigation of the *back stack* via deliberate control of *intents* that manage *activities*.
11. The application supports OS-related customization of *activities*, or of the *application* as a whole
(e.g. hide the OS status bar, keep the screen always on, show on the lock-screen).
12. The application integrates mobile features (e.g. vibrations, accelerometer, notifications, notification lights, GPS, flashlight, camera, audio, video).
13. The application can preserve state of its activities (e.g. in SQLite) when it gets terminated. This can be done automatically or upon a user request.
14. The application utilizes IPC to control another process or send data between processes.
15. The application can asynchronously upload or download contents from the Web (e.g. maintains the online Top Score player ranking/leaderboard).
16. The application allows for any kind of multiplayer experience across devices.
17. The application supports advanced 2D graphics features (e.g. transparency, gradients, shaders, complex animations).
18. The application supports basic 3D graphics instead or alongside 2D graphics.

19. The application is a “simulation” and as such it implements rules for systems with natural forces (e.g. gravity), imitating real-world processes (e.g. physical collisions), evolution of models of natural events (e.g. weather), etc. Note that an assignment that does not meet this requirement must still offer dynamic evolution of abstract behaviors (e.g. shapes changing colors, flying lines, swapping image sprites) without anchoring in physics, chemistry or biology.
20. The application supports one or more advanced features that are not present in this non-exhaustive list; together they count as a single feature towards the required 5 additional features.

Instructions

Submission

To complete the assignment, your team must write and test an Android application implementing the aforementioned requirements. The submission must be provided as a single ZIP archive including the following elements:

- **Source code** – a complete solution from Android Studio (Java or Kotlin) using the API level indicated above. The code must be compliable without warnings, and stable in execution to get full marks for the assignment. Before submission, clean up the code and exclude unnecessary files and artefacts that will be recreated during compilation.
- **Presentation slides** – a PDF document containing up to 5 pages or slides with:
  - Listing of team members,
  - Listing of required features and additional features implemented with a brief indication what they were used for (if not obvious from their role or context),
  - Technical highlights of the application related to the course contents.
  - General highlights – tell us what your team is proud of!
- **Video presentation** – a recording (maximum 3-5 minutes) containing a demonstration of the application and an overview of the structure of its source code or key ideas behind it. You can use Zoom/Microsoft Teams to record a quick session and use a camera to capture the screen of a mobile device or an Android simulator; you can also record a screen capture from the device directly and add a voiceover track. Participation of all team members is encouraged but not required.

Submit the code as a single *ZIP archive* via eLearn.

Grading

This assignment will contribute **12%** to the overall course grade.

The grading requirements are as follows:

- **Implementing required features** (25%)
- **Implementing additional features** (25%)
- **Correctness and stability** (20%)
- **Submitted presentation deliverables** (10%)
- **Code quality and consistent style** (10%)
- **Creativity and user experience** (10%)

Except for extraordinary situations, all team members will receive the same grade.

Late submissions will be graded one full letter grade down (e.g. from A- to B-) for each started period of 12 hours after the deadline.

Students may request for extensions should they provide valid formal reasons with documented evidence to justify their case. Such cases will be handled on a case-by-case basis. Request for extension after the deadline will be accepted only in extraordinary cases.

Honor code

SMU expects students to abide by the honor code and The SMU Student Code of Conduct. Specifically, students are expected to submit only their own work and not submit answers or code that have not been created by students themselves.

This is a team assignment. While you are encouraged to discuss with your classmates, submitted code must be your team's own work. No direct assistance from generative AI tools is allowed.

Artificial Intelligence

Artificial intelligence tools are great at helping experienced programmers build complex code faster. Experienced programmers have sufficient expertise to judge correctness and suitability of generated code.

But such tools also rob inexperienced programmers from meaningful learning experiences, making them reliable on accessibility of assistance, and taking away practice, repetition, dealing with mistakes, and developing deeper understanding of a programming process.

While your team is encouraged to discuss with your classmates and explore AI related suggestions, all submitted code must be your own work. No code generated from AI tools is allowed.

If you need to reach for AI to solve introductory problems, a problem beyond the reach of AI will definitely be beyond your reach.