NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade.

Programming-based puzzle video game

REPORT

A MINI PROJECT

REPORT

Submitted by

Akash.S

In partial fulfillment for the award of

the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade.

Certificate

This is to certify that the mini project work titled

Programming-based puzzle video game

Submitted in partial fulfillment of the degree of Bachelor of Engineering in Computer Science and Engineering

Submitted by

Akash.S

1NH16CS002

DURING

ODD SEMESTER 2019-2020

For

CSE76

Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner *Signature with date*

1._____ _____

2._____ _____

ABSTRACT

The player is presented with a series of puzzles that require them to program the nodes to perform specific actions on a set of numbers from one or more input terminals to produce pre-determined output at other terminals. For example, one task requires the player to double the value of the input at the output terminal. The game presents the list of inputs and the target output values that it is expecting and requires the players to develop the code for each node to match this; if during execution the output nodes receive unexpected outputs, the execution will cease and the player will have to rework their solution. Not all nodes are available in certain puzzles, so the player will need to route around these nodes. The game offers the player the ability to step through the execution of the code and insert debugging statements to determine logic issues within their code. Once the target output conditions are met, the player is considered to have solved the puzzle. The nodes that are disabled in puzzles contain cryptic messages related to the narrative of the game and which contain more information that can be accessed when the entire game is completed.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I thank the management, **Dr. Mohan Manghnani**, Chairman of NEW HORIZON EDUCATIONAL INSTITUTIONS for providing necessary infrastructure and creating good environment.

I also record here the constant encouragement and facilities extended to me by **Dr. Manjunatha**, Principal, NHCE, **Dr. Prashanth.C.S.R**, Dean Academics, **Dr. B. Rajalakshmi**, Head of the Department of Computer Science and Engineering. I extend my sincere gratitude to them.

I express my gratitude to **Mr. Muralidhara**, my project reviewer for constantly monitoring the development of the project and setting up precise deadlines. His valuable suggestions were the motivating factors in completing the work.

Finally a note of thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their cooperation extended to me and my friends, who helped me directly or indirectly in the course of the project work.

Akash S

1NH16CS002

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

To design and develop a fun and entertaining programming-based puzzle video game for android smartphones.

1.2 Objectives

- The puzzle has to be based on the player dragging and dropping programming instructions.
- The syntax and semantics of the programming language should be similar to assembly language.
- Every puzzle should have a clearly defined input and output which has to be met by the player to complete the puzzle.
- The game aims to be playable to a wide range of audience.
- Finally it has to be fun and entertaining.

1.3 Expected Outcomes

- A video game that look, plays and feels similar to a game called TIS-100.
- The player will solve a given problem in an assembly like language.
- There will be several machines that have to be programmed and they may have to work synchronously to produce the right result.
- There will be a list of in-built commands that the user can choose from. The operands to those commands are numbers.
- The game will come with a simple tutorial to explain an example program.

1.4 Hardware Requirements

Android OS 4.1 and above

Support for OpenGL ES 2.1

RAM of 1GB

CHAPTER 2

DataStructures and Android Platform

2.1 DataStructures

Data structures serve as the basis for abstract data types (ADT). The ADT defines the logical form of the data type. The data structure implements the physical form of the data type. Different types of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, relational databases commonly use B-tree indexes for data retrieval, while compiler implementations usually use hash tables to look up identifiers.

Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. Data structures can be used to organize the storage and retrieval of information stored in both main memory and secondary memory.

Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by a pointer—a bit string, representing a memory address, that can be itself stored in memory and manipulated by the program. Thus, the array and record data structures are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself.

The implementation of a data structure usually requires writing a set of procedures that create and manipulate instances of that structure. The efficiency of a data structure cannot be analyzed separately from those operations. This observation motivates the theoretical concept of an abstract data type, a data structure that is defined indirectly by the operations that may be performed on it,

and the mathematical properties of those operations (including their space and time cost).

There are numerous types of data structures, generally built upon simpler primitive data types:

- An array is a number of elements in a specific order, typically all of the same type (depending on the language, individual elements may either all be forced to be the same type, or may be of almost any type). Elements are accessed using an integer index to specify which element is required. Typical implementations allocate contiguous memory words for the elements of arrays (but this is not always a necessity). Arrays may be fixed-length or resizable.
- A linked list (also just called list) is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list. The principal advantage of a linked list over an array, is that values can always be efficiently inserted and removed without relocating the rest of the list. Certain other operations, such as random access to a certain element, are however slower on lists than on arrays.
- A record (also called tuple or struct) is an aggregate data structure. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called fields or members.
- A union is a data structure that specifies which of a number of permitted primitive types may be stored in its instances, e.g. float or long integer. Contrast with a record, which could be defined to contain a float and an integer; whereas in a union, there is only one value at a time. Enough space is allocated to contain the widest member datatype.
- A tagged union (also called variant, variant record, discriminated union, or disjoint union) contains an additional field indicating its current type, for enhanced type safety.
- An object is a data structure that contains data fields, like a record does, as well as various methods which operate on the data contents. An object is an in-memory instance of a class from a taxonomy. In the context of object-oriented programming, records are known as plain old data structures to distinguish them from objects.

In addition, graphs and binary trees are other commonly used data structures.

Most assembly languages and some low-level languages, such as BCPL (Basic Combined Programming Language), lack built-in support for data structures. On the other hand, many high-level programming languages and some higher-level assembly languages, such as MASM, have special syntax or other built-in support for certain data structures, such as records and arrays. For example, the C (a direct descendant of BCPL) and Pascal languages support structs and records, respectively, in addition to vectors (one-dimensional arrays) and multi-dimensional arrays.[11][12]

Most programming languages feature some sort of library mechanism that allows data structure implementations to be reused by different programs. Modern languages usually come with standard libraries that implement the most common data structures. Examples are the C++ Standard Template Library, the Java Collections Framework, and the Microsoft .NET Framework.

Modern languages also generally support modular programming, the separation between the interface of a library module and its implementation. Some provide opaque data types that allow clients to hide implementation details. Object-oriented programming languages, such as C++, Java, and Smalltalk, typically use classes for this purpose.

Many known data structures have concurrent versions which allow multiple computing threads to access a single concrete instance of a data structure simultaneously.

2.2 Android Platform

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September

2008. The current stable version is Android 10, released on September 3, 2019. The core Android source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License. This has allowed variants of Android to be developed on a range of other electronics, such as game consoles, digital cameras, PCs and others, each with a specialized user interface. Some well known derivatives include Android TV for televisions and Wear OS for wearables, both developed by Google.

Android's source code has been used as the basis of different ecosystems, most notably that of Google which is associated with a suite of proprietary software called Google Mobile Services (GMS), that frequently comes pre-installed on said devices. This includes core apps such as Gmail, the digital distribution platform Google Play and associated Google Play Services development platform, and usually apps such as the Google Chrome web browser. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google. Other competing Android ecosystems include Amazon.com's Fire OS, or LineageOS. Software distribution is generally offered through proprietary application stores like Google Play Store or Samsung Galaxy Store, or open source platforms like Aptoide or F-Droid, which utilize software packages in the APK format.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of December 2018, the Google Play Store features over 2.6 million apps.

Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language.Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support. The Go programming language is also supported, although with a limited set of application programming interfaces (API). In May 2017, Google announced support for Android app development in the Kotlin programming language.

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard. Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented, or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

Android devices boot to the homescreen, the primary navigation and information "hub" on Android devices, analogous to the desktop found on personal computers. Android homescreens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content, such as a weather forecast, the user's email inbox, or a news ticker directly on the homescreen. A homescreen may be made up of several pages, between which the user can swipe back and forth. Third-party apps available on Google Play and other app stores can extensively re-theme the homescreen, and even mimic the look of other operating systems, such as Windows Phone. Most manufacturers customize the look and features of their Android devices to differentiate themselves from their competitors.

Along the top of the screen is a status bar, showing information about the device and its connectivity. This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates. Notifications are "short, timely, and relevant information about your app when it's not in use", and when tapped, users are directed to a screen inside the app relating to the notification. Beginning with Android 4.1 "Jelly Bean", "expandable notifications" allow the user to tap an icon on the notification in order for it to expand and display

more information and possible app actions right from the notification. An All Apps screen lists all installed applications, with the ability for users to drag an app from the list onto the home screen. A Recents screen lets users switch between recently used apps.

2.3 SQLite

SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. Linking may be static or dynamic. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication.

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Due to the server-less design, SQLite applications require less configuration than client-server databases. SQLite is called zero-conf because it does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file system permissions given to the database file itself. Databases in client-server systems use file system permissions which give access to the database files only to the daemon process.

Another implication of the serverless design is that several processes may not be able to write to the database file. In server-based databases, several writers will all connect to the same daemon, which is able to handle its locks internally. SQLite on the other hand has to rely on file-system locks. It has less knowledge of the other processes that are accessing the database at the same time. Therefore, SQLite is not the preferred choice for write-intensive deployments. However, for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process.

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example, it partially provides triggers, and it cannot write to views (however it provides INSTEAD OF triggers that provide this functionality). While it provides complex queries, it still has limited ALTER TABLE function, as it cannot modify or delete columns.

SQLite uses an unusual type system for an SQL-compatible DBMS; instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is dynamically typed. Moreover, it is weakly typed in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if

the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added. Tables normally include a hidden rowid index column which gives faster access. If a database includes an Integer Primary Key column SQLite will typically optimize it by treating it as an alias for rowid, causing the contents to be stored as a strictly typed 64-bit signed integer and changing its behavior to be somewhat like an auto-incrementing column. Future versions of SQLite may include a command to introspect whether a column has behavior like that of rowid to differentiate these columns from weakly-typed, non-autoincrementing Integer Primary Keys.

SQLite with full Unicode function is optional.

Several computer processes or threads may access the same database concurrently. Several read accesses can be satisfied in parallel. A write access can only be satisfied if no other accesses are currently being serviced. Otherwise, the write access fails with an error code (or can automatically be retried until a configurable timeout expires). This concurrent access situation would change when dealing with temporary tables. This restriction is relaxed in version 3.7 when write-ahead logging (WAL) is turned on enabling concurrent reads and writes.

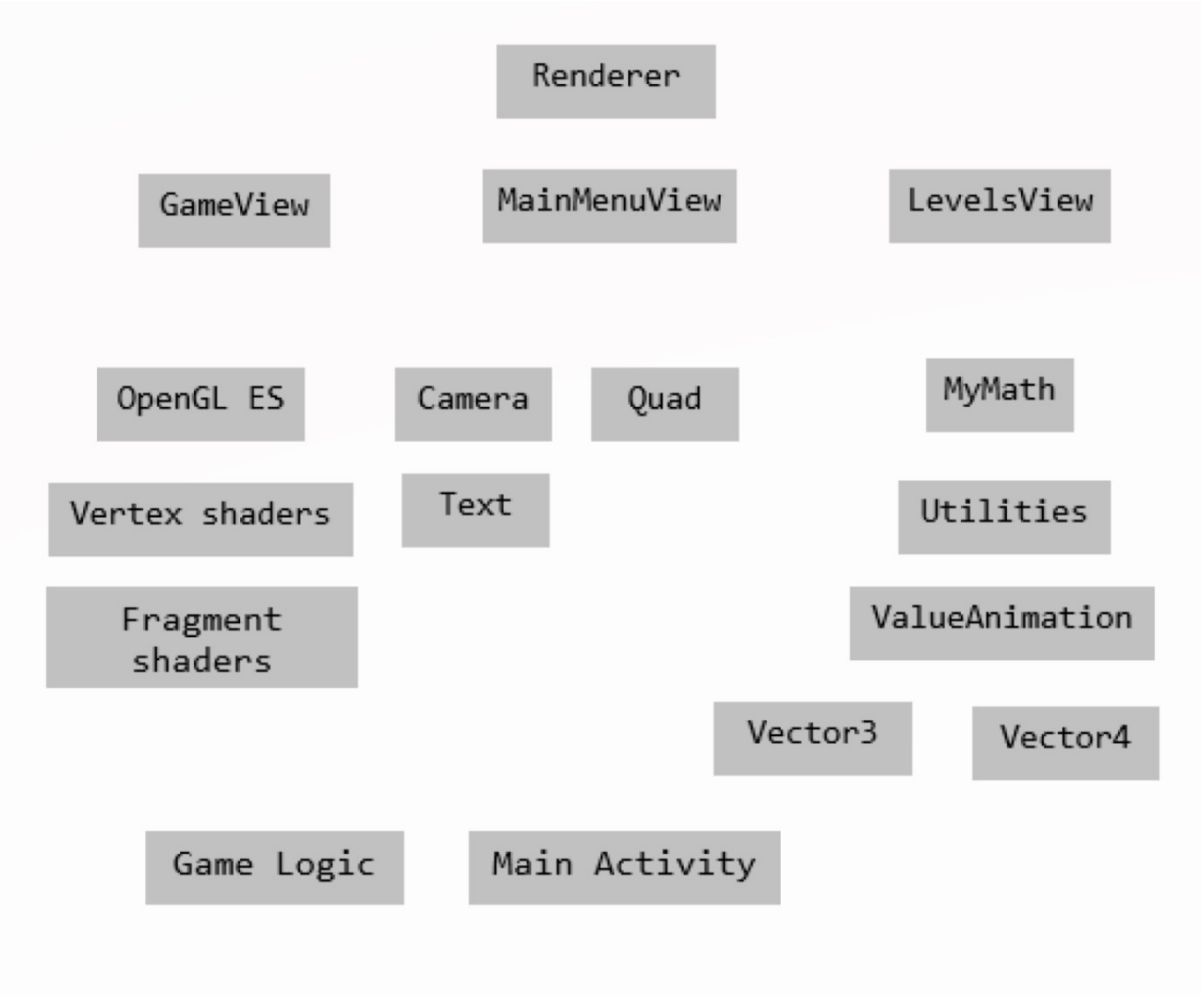
SQLite version 3.7.4 first saw the addition of the FTS4 (full text search) module, which features enhancements over the older FTS3 module. FTS4 allows users to perform full text searches on documents similar to how search engines search webpages. Version 3.8.2 added support for creating tables without rowid, which may provide space and performance improvements. Common table expressions support was added to SQLite in version 3.8.3.

In 2015, with the json1 extension and new subtype interfaces, SQLite version 3.9 introduced JSON content managing.

CHAPTER 3

Design

3.1 Architecture



3.2 Algorithm

1) Initialise the MainActivity, set the renderer to OpenGL renderer.

2) Load the textures, cameras and levels.

3) Display the MainMenu.

4) Select NewGame, Continue Game or Options

5) Select any of the available levels.

6) Write a program to meet the objectives of the program using the correct syntax.

7) Run the program to verify the outputs.

8) Make the program efficient. Goto step 6.

9) Goto step 5.

10) Exit.

- A Game loop:

  The **game loop** is the overall flow control for the entire game program. It's a loop because the game keeps doing a series of actions over and over again until the user quits. Each iteration of the game loop is known as a **frame**. Most real-time games update several times per second: 30 and 60 are the two most common intervals. If a game runs at 60 FPS (**frames per second**), this means that the game loop completes 60 iterations every second.

  At a high level, a basic game loop might look like this:

  While game is running:

  Process inputs

  Update game world

  Generate outputs

  End while

CHAPTER 4

Implementation

4.1 Android concepts used

- Activities - An *activity* is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app:

    - Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
    - Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
    - Helping the app handle having its process killed so the user can return to activities with their previous state restored.
    - Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

- Sensors - Android devices typically have a set of *sensors* built in which you can access from your Android applications. For instance, the built-in GPS in many smart phones is a sensor. Thus, you can get access to the GPS from inside your Android applications.
- App resources - An Android app is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the app. For example, you can define

animations, menus, styles, colors, and the layout of activity user interfaces with XML files. Using app resources makes it easy to update various characteristics of your app without modifying code. Providing sets of alternative resources enables you to optimize your app for a variety of device configurations, such as different languages and screen sizes. For every resource that you include in your Android project, the SDK build tools define a unique integer ID, which you can use to reference the resource from your app code or from other resources defined in XML. For example, if your app contains an image file named logo.png (saved in the res/drawable/ directory), the SDK tools generate a resource ID named R.drawable.logo. This ID maps to an app-specific integer, which you can use to reference the image and insert it in your user interface.

- OpenGL ES - OpenGL for Embedded Systems (OpenGL ES or GLES) is a subset of the OpenGL computer graphics rendering application programming interface (API) for rendering 2D and 3D computer graphics such as those used by video games, typically hardware-accelerated using a graphics processing unit (GPU). It is designed for embedded systems like smartphones, tablet computers, video game consoles and PDAs. OpenGL ES is the "most widely deployed 3D graphics API in history".

  The API is cross-language and multi-platform. The libraries GLUT and GLU are not available for OpenGL ES. OpenGL ES is managed by the non-profit technology consortium Khronos Group. Vulkan, a next-generation API from Khronos, is made for simpler high performance drivers for mobile and desktop devices.

  OpenGL ES 2.0 was publicly released in March 2007. It is roughly based on OpenGL 2.0, but it eliminates most of the fixed-function rendering pipeline in favor of a programmable one in a move similar to the transition from OpenGL 3.0 to 3.1. Control flow in shaders is generally limited to forward branching and to loops where the maximum number of iterations can easily be determined at compile time. Almost all rendering features of the transform and lighting stage, such as the specification of materials and light parameters formerly specified by the fixed-function API, are replaced by shaders written by the graphics programmer. As a

result, OpenGL ES 2.0 is not backward compatible with OpenGL ES 1.1. Some incompatibilities between the desktop version of OpenGL and OpenGL ES 2.0 persisted until OpenGL 4.1, which added the GL_ARB_ES2_compatibility extension. Actual version is 2.0.25.

The libraries GLUT and GLU are not available for OpenGL ES. OpenGL ES is managed by the non-profit technology consortium Khronos Group. Vulkan, a next-generation API from Khronos, is made for simpler high performance drivers for mobile and desktop devices.

OpenGL® ES is a royalty-free, cross-platform API for rendering advanced 2D and 3D graphics on embedded and mobile systems - including consoles, phones, appliances and vehicles. It consists of a well-defined subset of desktop OpenGL suitable for low-power devices, and provides a flexible and powerful interface between software and graphics acceleration hardware.

OpenGL ES API Versions at a Glance

OpenGL ES 3.2 - Additional OpenGL functionality

The latest in the series, OpenGL ES 3.2 added additional functionality based on the Android Extension Pack for OpenGL ES 3.1, which brought the mobile API's functionality significantly closer to it's desktop counterpart - OpenGL.

OpenGL ES 3.1 - Bringing Compute to Mobile Graphics

Despite being only a bump in the minor revision of the API, OpenGL ES 3.1 was an enormous milestone for the API, as it added the ability to do general purpose compute in the API, bringing compute to mobile graphics.

OpenGL ES 3.0 - Enhanced Graphics

OpenGL ES 3.0 was another evolutionary step for OpenGL ES, notably including multiple render targets, additional texturing capabilities, uniform buffers, instancing and transform feedback.

OpenGL ES 2.0 - Programmable Shading

OpenGL ES 2.0 was the first portable mobile graphics API to expose programmable shaders in the then latest generation of graphics hardware. It remains a prevalent API today, and still is the most widely available 3D graphics API, and remains a solid choice to target the widest range of devices in the market.

OpenGL ES 1.X - Fixed Function Graphics

OpenGL ES 1.0 and 1.1 were the first portable mobile graphics APIs, defined relative to the OpenGL 1.5 specification, providing fixed function graphics acceleration

- Apps and Packages - Given a bucket of source code and a basket of resources, the Android build tools will give you an application as a result. The application comes in the form of an APK file. It is APK file that you will upload to the Play Store or distribute by other means.

  Important thing to learn is that each android application has a unique package name and it must fulfill three requirements:

  1. It must be a valid java package name, as some java source code will be generated by the android build tools in this package.
  2. No two applications can exist on a device at the same time with the same package.
  3. No two applications can be uploaded to the Play Store having the same package.
  4. Package objects contain version information about the implementation and specification of a Java package. This versioning information is retrieved and made available by the ClassLoader instance that loaded the class(es). Typically, it is stored in the manifest that is distributed with the classes.
  5. The set of classes that make up the package may implement a particular specification and if so the specification title, version number, and vendor strings identify that specification. An application can ask if the package is compatible with a particular version, see the isCompatibleWith method for details.

4.2 Functionality of the project

- It teaches you how to program in an assembly like language.
- The puzzles can vary from easy to hard.
- A result of how well your program performed after each execution.

4.3 SQLite Database Connectivity

SQLite is :

◦ Open-source

◦ Standards-compliant

◦ Lightweight

◦ Single-tier

◦ Extremely reliable

It has been implemented as a compact C library that's included as part of the Android software stack. Each SQLite database is an integrated part of the application that created it. This reduces external dependencies, minimizes latency, and simplifies transaction locking and synchronization.

SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. It is embedded in android bydefault. So, there is no need to perform any database setup or administration task. Here, we are going to see the example of sqlite to store and fetch the data. Data is displayed in the logcat. For displaying data on the spinner or listview, move to the next page. SQLiteOpenHelper class provides the functionality to use the SQLite database.

The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management. For performing any database operation, you have to

provide the implementation of onCreate() and onUpgrade() methods of SQLiteOpenHelper class.

a) Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);

b) **Database – Insertion** - We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below.

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username VARCHAR,Password VARCHAR);");

mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");

c) Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);

resultSet.moveToFirst();

String username = resultSet.getString(0);

String password = resultSet.getString(1);

Design:

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead,

the SQLite library is linked in and thus becomes an integral part of the application program. Linking may be static or dynamic. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication.

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Due to the server-less design, SQLite applications require less configuration than client-server databases. SQLite is called zero-conf because it does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file system permissions given to the database file itself. Databases in client-server systems use file system permissions which give access to the database files only to the daemon process.

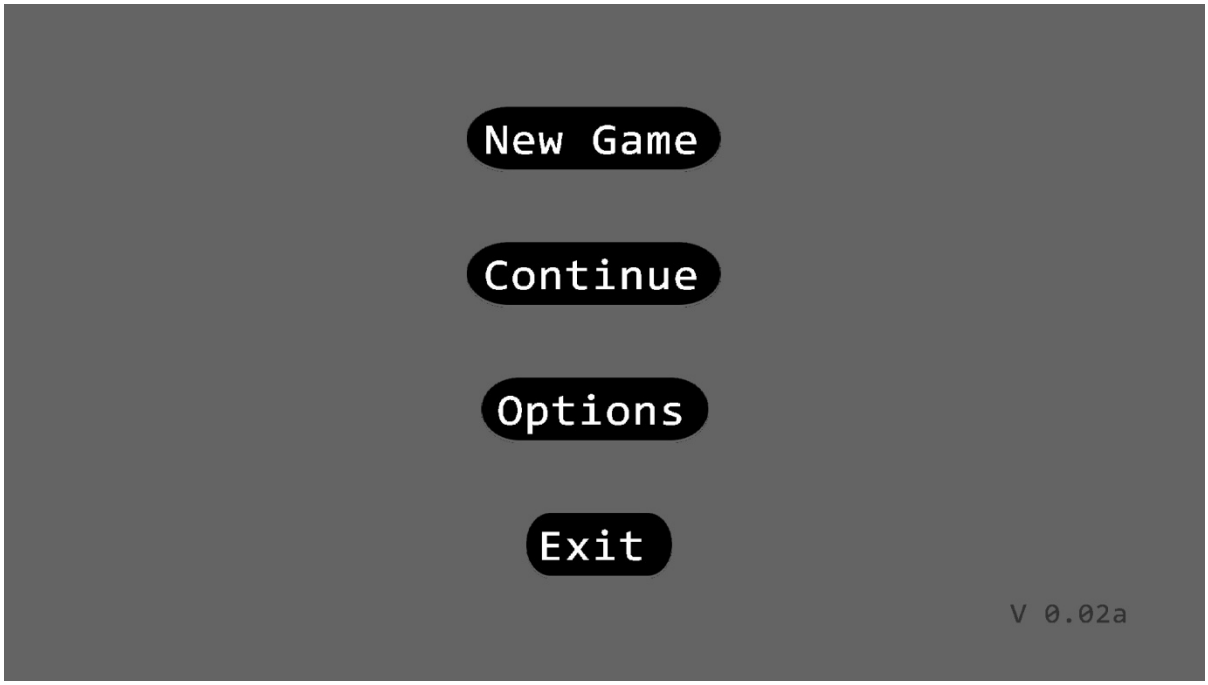
Another implication of the serverless design is that several processes may not be able to write to the database file. In server-based databases, several writers will all connect to the same daemon, which is able to handle its locks internally. SQLite on the other hand has to rely on file-system locks. It has less knowledge of the other processes that are accessing the database at the same time. Therefore, SQLite is not the preferred choice for write-intensive deployments. However, for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process.

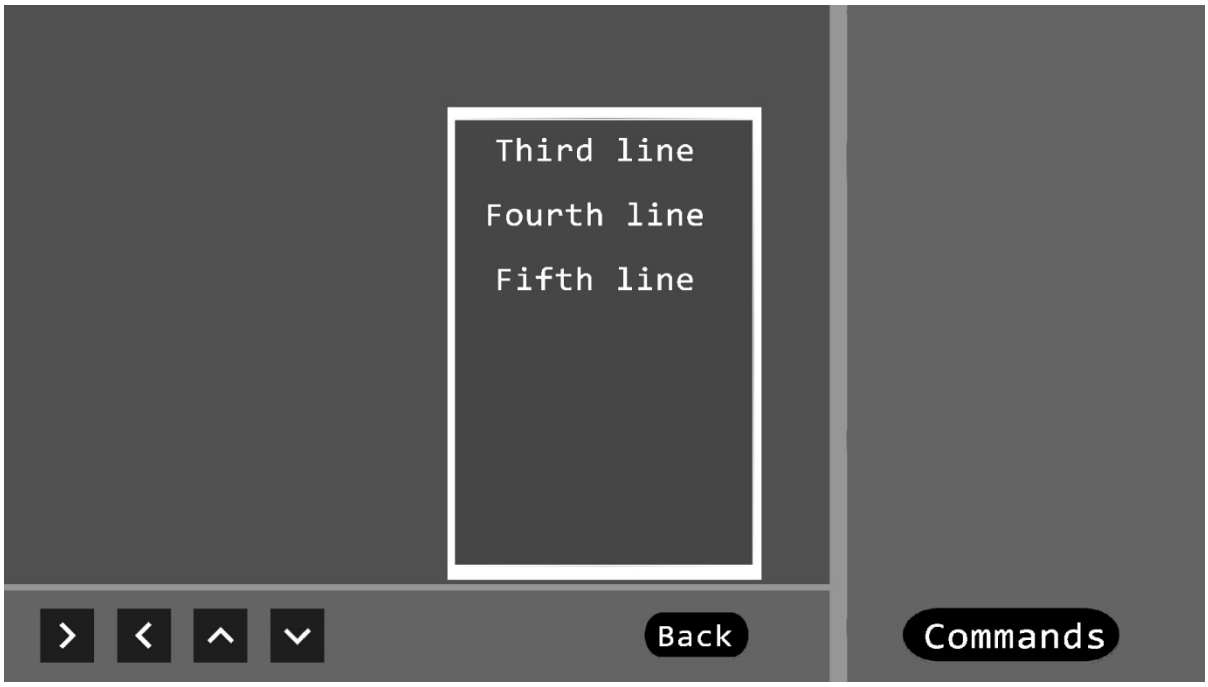
SQLite uses PostgreSQL as a reference platform. "What would PostgreSQL do" is used to make sense of the SQL standard. One major deviation is that, with the exception of primary keys, SQLite does not enforce type checking; the type of a value is dynamic and not strictly constrained by the schema (although the schema will trigger a conversion when storing, if such a conversion is potentially reversible). SQLite strives to follow Postel's Rule.

CHAPTER 5

Sample Output

Note: These images were taken when the game was still in development(incomplete).





CHAPTER 6

Conclusion

Puzzle video games make up a genre of video games that emphasize puzzle solving. The types of puzzles can test many problem-solving skills including logic, pattern recognition, sequence solving, and word completion. The player may have unlimited time or infinite attempts to solve a puzzle, or there may be a time limit, or simpler puzzles may be made difficult by having to complete them in real time, as in Tetris.

The genre is very broad, but it generally involves some level of abstraction and may make use of colors, shapes, numbers, physics, or complex rules. Unlike many video games, puzzle video games often do make use of "lives" that challenge a player by limiting the number of tries. In puzzle video games, players often try for a high score or to progress to the next level by getting to a certain place or achieving some criteria.

Puzzle games focus on logical and conceptual challenges, although often the games add time-pressure or other action-elements. Although many action games and adventure games involve puzzles such as obtaining inaccessible objects, a true puzzle game focuses on puzzle solving as the primary gameplay activity. Games usually involve shapes, colors, or symbols, and the player must directly or indirectly manipulate them into a specific pattern.

Rather than presenting a random collection of puzzles to solve, puzzle games typically offer a series of related puzzles that are a variation on a single theme. This theme could involve pattern recognition, logic, or understanding a process. These games usually have a simple set of rules, where players manipulate game pieces on a grid, network or other interaction space. Players must unravel clues in order to achieve some victory condition, which will then allow them to advance to the next level. Completing each puzzle will usually lead to a more difficult challenge, although some games avoid exhausting the player by offering easier levels between more difficult ones.

In adventure games, some stages require solving puzzles as a way to advance the story.

There is a large variety of puzzle game types. Some feed to the player a random assortment of blocks or pieces that they must organize in the correct manner, such

as Tetris, Klax and Lumines. Others present a preset game board or pieces and challenge the player to solve the puzzle by achieving a goal (Bomberman, The Incredible Machine).

Puzzle games are often easy to develop and adapt, being implemented on dedicated arcade units, home video game consoles, personal digital assistants, and mobile phones.

Action puzzle

An action puzzle or arcade puzzle requires that the player manipulates game pieces in a real-time environment, often on a single screen and with a time limit, to solve the puzzle or clear the level. This is a broad term that has been used to describe several subsets of puzzle game. Firstly, it includes falling-block puzzles such as Tetris and KLAX. It includes games with characters moving through an environment, controlled either directly (Lode Runner) or indirectly (Lemmings). This can cross-over with other action genres: a platform game which requires a novel mechanic to complete levels might be a "puzzle platformer", such as manipulating time in Braid. Finally, it includes other action games that require timing and accuracy with pattern-matching or logic skills, such as the first-person Portal and The Talos Principle.

Other notable action puzzle games include Team Ico's Ico, a linear, story driven game with puzzles based around traversing puzzle environments while protecting a helpless companion. Also made by Team Ico is Shadow of the Colossus, a game in which the player solves puzzles that involve finding and exploiting the weaknesses of giant beasts in combat. Nintendo's The Legend of Zelda: Breath of the Wild is another example of an action puzzle game, the primary objective being to seek out and solve physics-based puzzles which offer helpful upgrades for defeating the final boss.

Hidden object game

A hidden object game (sometimes called hidden picture) is a genre of puzzle video game in which the player must find items from a list that are hidden within a picture. Hidden object games are a popular trend in casual gaming, and are comparatively inexpensive to buy. Time-limited trial versions of these games are usually available for download.

An early hidden object game was Alice: An Interactive Museum. Computer Gaming World reported in 1993 that "one disadvantage of searching through screen after screen for

'switches' is that after a while one develops a case of 'clickitus' of the fingers as one repeatedly punches that mouse button like a chicken pecking at a farmyard". Other early incarnations are the video game adaptations of the I Spy books published by Scholastic Corporation since 1997.

Publishers of hidden object games include Sandlot Games, Big Fish Games, Awem Studio, SpinTop Games, and Codeminion. Examples of hidden object game series include Awakening, Antique Road Trip (both by Boomzap Entertainment), Dream Chronicles (PlayFirst), Mortimer Beckett (RealArcade/GameHouse), Mystery Trackers (by Elephant games), Hidden Expedition and Mystery Case Files (both by Big Fish Games).

Reveal the picture game

A reveal the picture game is a type of puzzle game that features piece-by-piece revealing of a photo or picture. A free online example is PicTAPr, which divides an image into 16 square pieces.

Physics game

A physics game is a type of puzzle video game wherein the player must use the game's physics to complete each puzzle. Physics games use realistic physics to make games more challenging. The genre is especially popular in online flash games and mobile games. Educators have used these games to demonstrate principles of physics.

Popular physics games include The Incredible Machine, World of Goo, Crayon Physics Deluxe, Angry Birds, Cut the Rope, Peggle, Portal, Portal 2, Monster Strike and The Talos Principle.

In tile-matching video games, the player manipulates tiles in order to make them disappear according to a matching criterion. The genre began with 1985's Chain Shot!. It includes games of the "falling block" variety such as Tetris, games that require pieces to be swapped such as Bejeweled or Candy Crush Saga, games that adapt the classic tile-based game Mahjong such as Mahjong Trails, and games in which are pieces are shot on the board such as Zuma. In many recent tile-matching games, the matching criterion is to place a given

number of tiles of the same type so that they adjoin each other. That number is often three, and the corresponding subset of tile-matching games is referred to as "match-three games."

There have also been many digital adaptations of traditional puzzle games, including solitaire and mahjong solitaire. Even familiar word puzzles, number puzzles, and association puzzles have been adapted as games such as Dr. Kawashima's Brain Training.

Mobile Gaming's Present: 2017-2019

Most notably, the rise of battle royale games like Fortnite and PUBG Mobile proved that technology had finally advanced to the point where the mobile gaming experience between console/PC and mobile was seamless, and the rewards for developers were enormous. As of the summer of 2018, Fortnite alone was pulling in well over $100 million a month, and that's just direct revenue for the publisher.

But while battle royale games rightfully continue to garner big headlines, they're one story among many in the mobile gaming world. The past year has seen new blockbuster titles like Harry Potter: Wizards Unite and Jurassic World Alive that signal the industry's revived investment in the AR technology that made Pokémon Go such a breakaway success.

Mobile Gaming's Future: 2018 and beyond

Now that we've laid out the road behind us and our current location, it's time to finally turn our attention toward the future. For this series, we'll talk about the foreseeable future, as in the next five years. Given the staggering pace of technological advancement, any predictions beyond that time frame would just be too uncertain to be actionable.

With that said, the evolution of data science has made forecasting a more exact science than ever. Pair this with industry expertise and knowledge of major upcoming trends like the rollout of 5G, and the future can be put clearly into focus. That future will be one that sees gaming continuing to catapult off advancements in hardware and networks that, along with other advancements like AI, will offer highly personalized experiences that bring mobile games into every moment of our lives.

Overall, growth in maturing and emerging markets will help drive mobile gaming to even greater heights as games are projected to be the highest category for consumer spend into 2023.

Puzzle games used to be a simple genre with a simple idea. You solve puzzles for time killing enjoyment. However, the genre ballooned in a big way on mobile. In fact, it's one of the most popular genres on the entire platform. It started with simple viral games like 2048 and evolved into genuinely enjoyable experiences like Monument Valley and Telltale Games titles (until they went under). These days, you can find some seriously good puzzle games on Android. Gone are the days of super simple mechanics and graphics with goofy ideas. Some of these games are actually fairly intense and complex with stories and even really good graphics.

Here is a list of best puzzle video games:

- Portal
- Tetris
- Portal 2
- The Witness
- Braid
- Myst
- Limbo
- World of Goo
- Inside
- Threes
- Monument valley
- Fez
- The Talos principle