

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI**



Mini Project Report on

“BLOCK-CHAIN SIMULATION”

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

ARUN SHENOY

USN: 1BY17CS032

CHARAN KALSHETTY

USN: 1BY17CS041

Under the guidance of

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.

2019-2020

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Project work entitled “**BLOCK-CHAIN SIMULATION**” is a bona fide work carried out by **Arun Shenoy (1BY17CS032)** and **Charan Kalshetty (1BY17CS041)** in partial fulfillment for *Mini Project* during the year 2019-2020. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

**Signature of the
Guide with date**
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

**Signature of HOD
with date**
Dr. Anil G N
Prof & Head
CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. ANIL G N, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

ARUN SHENOY (1BY17CS032)

CHARAN KALSHETTY (1BY17CS041)

ABSTRACT

A block-chain, originally block chain, is a growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.

By design, a block chain is resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". For use as a distributed ledger, a block chain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. Although block chain records are not unalterable, block chains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance. Decentralized consensus has therefore been claimed with a block chain.

With the help of computer graphics we can create the simulation process. We can create a graphical user interface which is the simulator and the processes achieved by the interface is the process or the simulation.

A simulation is the best way to understand any concept. It aids in the interaction between the user and an actual process (simulation) in form of a program (simulator). It acts as an interface for the user which he/she can interact with, to better understand the process.

TABLE OF CONTENTS

1. ACKNOWLEDGEMENT

2. ABSTRACT

3. TABLE OF CONTENTS

CHAPTER NO. TITLE	PAGE NO
CHAPTER 1 INTRODUCTION	01
1.1 Brief Introduction	01
1.2 Motivation	02
1.3 Scope	02
1.4 Problem Statement	02
1.5 Proposed System	03
1.6 Limitations	03
CHAPTER 2 LITERATURE SURVEY	04
CHAPTER 3 SYSTEM REQUIREMENT	05
CHAPTER 4 SYSTEM ANALYSIS	06
CHAPTER 5 IMPLEMENTATION	08
CHAPTER 6 INTERPRETATION OF RESULTS	13
CONCLUSION	21
FUTURE ENHANCEMENTS	21
BIBLIOGRAPHY	22

CHAPTER 1

INTRODUCTION

1.1 BRIEF INTRODUCTION

A block-chain is a decentralized, distributed, and oftentimes public, digital ledger that is used to record transactions across many computers so that any involved record cannot be altered retroactively, without the alteration of all subsequent blocks. This allows the participants to verify and audit transactions independently and relatively inexpensively. A block-chain database is managed autonomously using a peer-to-peer network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests. Such a design facilitates robust workflow where participants' uncertainty regarding data security is marginal. The use of a block-chain removes the characteristic of infinite reproducibility from a digital asset. It confirms that each unit of value was transferred only once, solving the long-standing problem of double spending. A block-chain has been described as a value-exchange protocol. A block-chain can maintain title rights because, when properly set up to detail the exchange agreement, it provides a record that compels offer and acceptance.

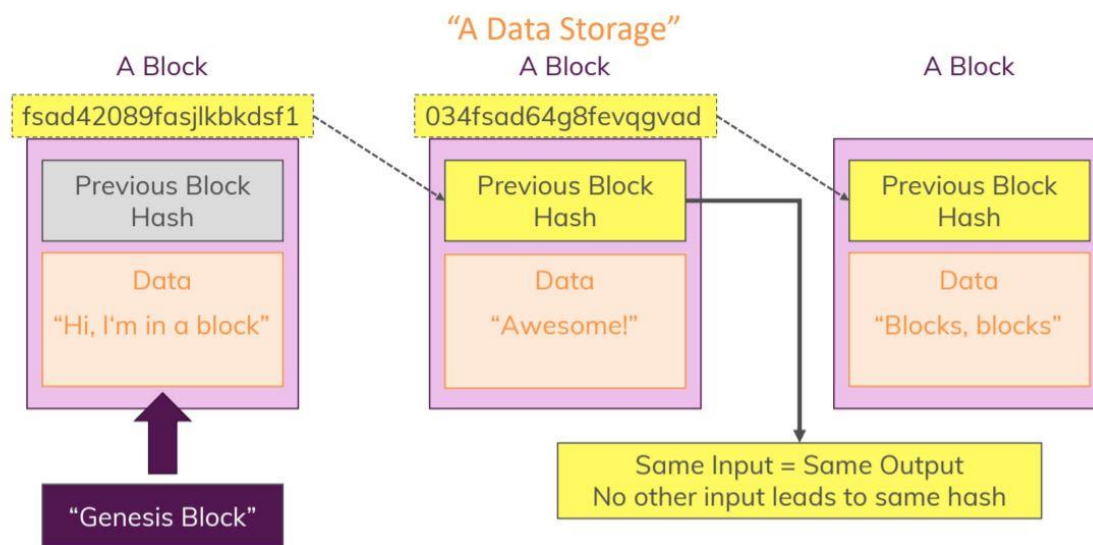


Fig: Simple Block-chain

Blocks:

Blocks hold batches of valid transactions that are hashed and encoded into a Merkle tree. Each block includes the cryptographic hash of the prior block in the block-chain, linking the

BLOCK-CHAIN SIMULATION

two. The linked blocks form a chain. This iterative process confirms the integrity of the previous block, all the way back to the original genesis block.

Decentralization:

Peer-to-peer block-chain networks lack centralized points of vulnerability that computer crackers can exploit; likewise, it has no central point of failure. Block-chain security methods include the use of public-key cryptography. A public key (a long, random-looking string of numbers) is an address on the block-chain. Value tokens sent across the network are recorded as belonging to that address. A private key is like a password that gives its owner access to their digital assets or the means to otherwise interact with the various capabilities that block-chains now support. Data stored on the block-chain is generally considered incorruptible.

1.2 MOTIVATION

Since block-chain is a new and emerging technology it is hard for people, sometimes even computer scientists to understand the exact working of the block-chain and the best way to understand something is to visualize it. We hope to make use of the concepts learned in computer graphics to help people visualize and clearly understand the working of a block-chain. This includes how blocks are mined, how transaction data is stored in blocks and how these blocks themselves are linked to each other to form a block-chain.

1.3 SCOPE

A graphical interface made using computer graphics concepts, which is a simulation of the block-chain process. It helps the user to better understand the block-chain process. The simulation graphically represents many block-chain concepts. This helps the user get familiar to many concepts theoretically as well as graphically. It is not an exact block-chain process but a representation of it.

1.4 PROBLEM STATEMENT

Create a graphical interface which acts as a simulator to simulate block-chain concepts to help the user to better understand and get familiarised with the same.

1.5 PROPOSED SYSTEMS

With the help of OpenGL and C++ we created a graphical interface for the user to interact with it and understand various Block-chain concepts. The graphical interface is a simulation of the Block-chain process.

1.6 LIMITATION

The simulation created here is not the actual block-chain process but just a representation of it. It does not actually perform any of the block-chain functions. It is just a graphical representation of the block-chain process which is interactive with the user.

CHAPTER 2**LITERATURE SURVEY**

Computer graphics is branch of computer science that deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer displays, and many specialized applications. A great deal of specialized hardware and software has been developed, with the displays of most devices being driven by computer graphics hardware. It is a vast and recently developed area of computer science. The phrase was coined in 1960 by computer graphics researchers Verne Hudson and William Fetter of Boeing. It is often abbreviated as CG, or typically in the context of film as CGI.

Some topics in computer graphics include user interface design, sprite graphics, rendering, ray tracing, geometry processing, computer animation, vector graphics, 3D modelling , shaders , GPU design, implicit surface visualization, image processing, computational photography, scientific visualization, computational geometry and computer vision, among others. The overall methodology depends heavily on the underlying sciences of geometry, optics, physics, and perception.

We use OpenGL for implementing the computer graphics concepts. OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

With the help of the above we are creating a graphical interface for the use which is interactive. This interface is like an interactive simulation. A simulation is an approximate imitation of the operation of a process or system that represents its operation over time. Simulation is used in many contexts, such as simulation of technology for performance tuning or optimizing, safety engineering, testing, training, education, and video games. We are using it to implement Block-chain concepts to help users understand the various functions and concepts related to block-chain. It not only helps them understand but also visualise the block-chain process.

CHAPTER 3**SYSTEM REQUIREMENTS****HARDWARE REQUIREMENTS:**

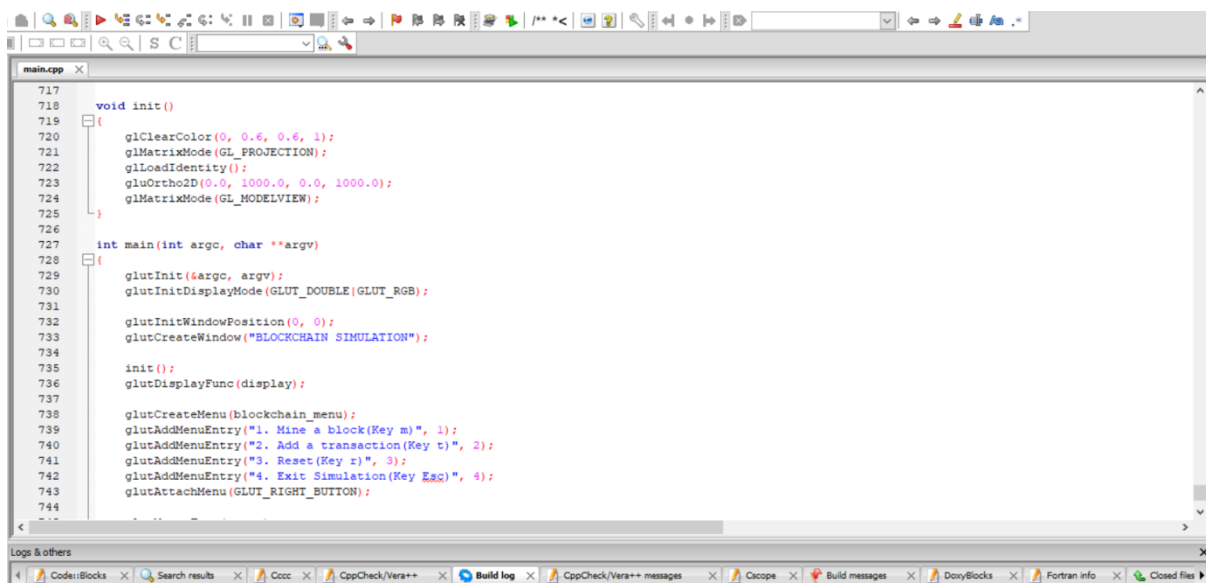
1. A laptop or pc running Windows or Linux.
2. Minimum 4 GB RAM.
3. Minimum of 5 GB free space of memory.

SOFTWARE REQUIREMENTS:

1. Code-Blocks v17.12 or above.
2. Glut library
3. C++ compiler

SYSTEM ANALYSIS

Our system consists of two main parts that is the OpenGL backend process for the graphical animation displayed to the end user and the block-chain logic needed for simulation. The OpenGL takes in various inputs from the user like clicking a mouse button and produces an output which changes the elements in the graphic interface.



```

717
718 void init()
719 {
720     glClearColor(0, 0.6, 0.6, 1);
721     glMatrixMode(GL_PROJECTION);
722     glLoadIdentity();
723     gluOrtho2D(0.0, 1000.0, 0.0, 1000.0);
724     glMatrixMode(GL_MODELVIEW);
725 }
726
727 int main(int argc, char **argv)
728 {
729     glutInit(&argc, argv);
730     glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
731
732     glutInitWindowPosition(0, 0);
733     glutCreateWindow("BLOCKCHAIN SIMULATION");
734
735     init();
736     glutDisplayFunc(display);
737
738     glutCreateMenu(blockchain_menu);
739     glutAddMenuEntry("1. Mine a block(Key m)", 1);
740     glutAddMenuEntry("2. Add a transaction(Key t)", 2);
741     glutAddMenuEntry("3. Reset(Key r)", 3);
742     glutAddMenuEntry("4. Exit Simulation(Key Esc)", 4);
743     glutAttachMenu(GLUT_RIGHT_BUTTON);
744
745     return 0;
746 }
  
```

Fig: Window Level GLUT code



Fig: Graphical User Interface

BLOCK-CHAIN SIMULATION

Whereas the core is maintained by the block-chain part of the code.

```

442     }
443
444     //Mine a block
445
446     if (what_to_do == 1)
447     {
448         if (manip_block != -2)
449         {
450             what_to_do = 0;
451             printf("\n");
452             return;
453         }
454         if (no_of_blocks == 14)
455         {
456             what_to_do = 0;
457             printf("\n");
458             return;
459         }
460         add_block();
461         what_to_do = 0;
462     }
463
464     //Add open transaction
465
466     if (what_to_do == 2)
467     {
468         if (no_of_ot == 14)
469         {
470             what_to_do = 0;
471             printf("\n");
472             return;
473         }
474         x = x + open_transaction_width + padding;
475         no_of_ot += 1;
476         what_to_do = 0;
477     }

```

Fig: Some Block-chain operations

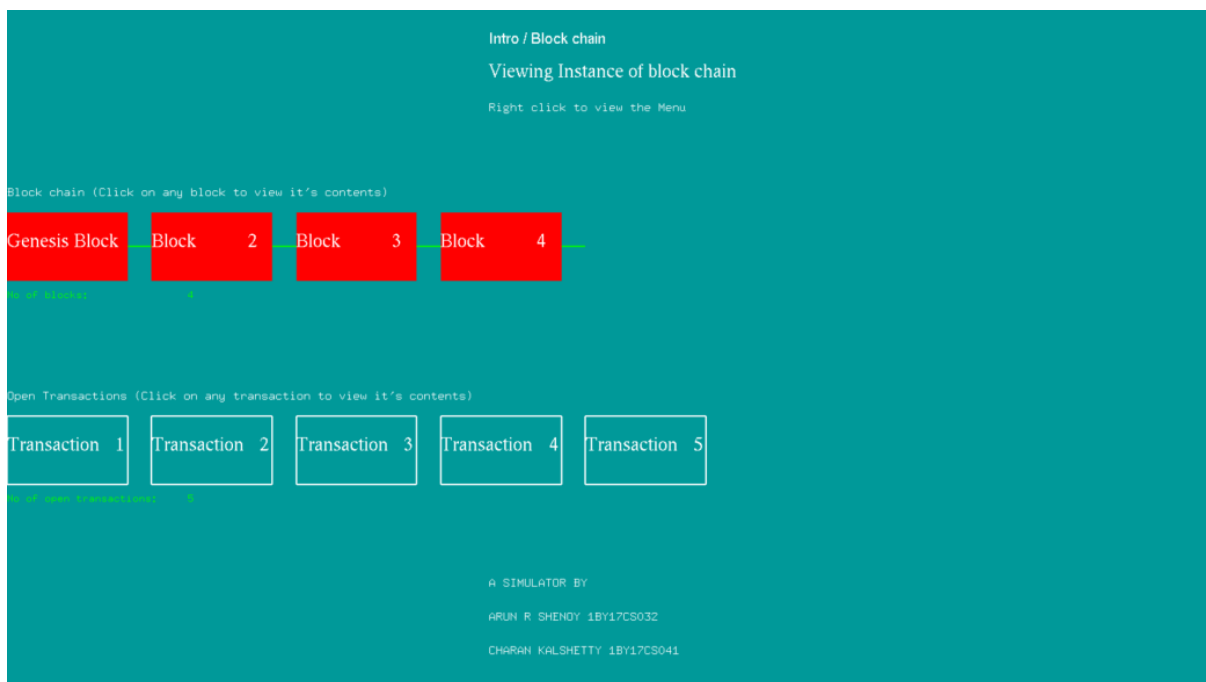


Fig: Block-chain

CHAPTER 5

IMPLEMENTATION**Function for the Introduction Page**

Gives the beginning page of the simulation.

Uses a function drawStrokeText which is implemented to take a string and render the characters in the string using glutStrokeCharacter().

The Fig: Graphical User Interface above shows the output of this code

```
void intro(int x, int y)
{
    RenderString(x + 50, y + 350, "Intro ", GLUT_BITMAP_HELVETICA_18);
    drawStrokeText(x - 250, y + 200, "COMPUTER GRAPHICS LAB MINI PROJECT", 2);
    drawStrokeText(x - 100, y, "BLOCKCHAIN SIMULATOR", 3);
    drawStrokeText(x - 100, y - 100, "Press N to create a new simulation instance", 2, 0, 1, 0, 0.15f, 0.15f);
    drawStrokeText(x + 50, y - 400, "A SIMULATOR BY", 2, 1, 1, 1, 0.2f, 0.2f);
    drawStrokeText(x - 300, y - 500, "ARUN R SHENOY 1BY17CS032", 2, 1, 1, 1, 0.2f, 0.2f);
    drawStrokeText(x + 150, y - 500, "CHARAN KALSHETTY 1BY17CS041", 2, 1, 1, 1, 0.2f, 0.2f);
}

void drawStrokeText(int x, int y, char*string, float w = 1, float r = 1, float g = 1, float b = 1, float sx = 0.3f, float sy = 0.3f)
{
    int s = 0;
    char *c;
    glColor3f(r, g, b);
    glLineWidth(w);
    glPushMatrix();
    glTranslatef(x, y+s, s);
    glScalef(sx, sy, s);
    for (c=string; *c != '\0'; c++)
    {
        glutStrokeCharacter(GLUT_STROKE_ROMAN, *c);
    }
    glPopMatrix();
}
```

BLOCK-CHAIN SIMULATION

Function used to display block-chain

To draw the blocks as a polygon.

```
void display_blocks()
{
    RenderString(400, 900, "Viewing Instance of block chain", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(400, 950, "Intro / Block chain ", GLUT_BITMAP_HELVETICA_18);
    RenderString(400, 980, "Right click to view the Menu", GLUT_BITMAP_9_BY_15);
    int tempx = 0;
    char str[3];

    RenderString(0, y+125, "Block chain (Click on any block to view it's contents)", GLUT_BITMAP_9_BY_15);

    if (no_of_blocks >= 9)
    {
        block_width = 50;
        block_height = 100;
        block_font = GLUT_BITMAP_TIMES_ROMAN_10;
    }

    for (int i = 1; i <= no_of_blocks; i++)
    {
        if (i <= 9)
        {
            to_string_digit(str, i);
        }
        else
        {
            to_string_digits(str, i);
        }
        if (i >= manip_block && manip_block != -2)
        {
            draw_polygon(tempx, y, block_width, block_height, 0.0, 0.0, 0.0);
        }
        else
        {
            draw_polygon(tempx, y, block_width, block_height);
        }
        if (i == 1)
        {
            RenderString(tempx, y + (block_height/2), "Genesis Block", block_font);
        }
        else
        {
            RenderString(tempx, y + (block_height/2), "Block", block_font);
            RenderString(tempx + block_width - 20, y + (block_height/2), str, block_font);
        }
        draw_line(tempx + block_width, y+(block_height/2), tempx + block_width + padding, y+(block_height/2));
        tempx = tempx + block_width + padding;
    }

    RenderString(0, y - 25, "No of blocks: ", GLUT_BITMAP_9_BY_13, 0, 1, 0);
    RenderString(150, y - 25, str, GLUT_BITMAP_9_BY_13, 0, 1, 0);
    if (manip_block != -2)
    {
        RenderString(0, y - 50, "Blockchain is invalid please press C to contact peer nodes and replace it", GLUT_BITMAP_9_BY_13, 1, 0, 0);
    }
    RenderString(400, 150, "A SIMULATOR BY", GLUT_BITMAP_9_BY_15);
    RenderString(400, 100, "ARUN R SREMOY 1BY17CS032", GLUT_BITMAP_9_BY_15);
    RenderString(400, 50, "CHARAN KALSHETTY 1BY17CS041", GLUT_BITMAP_9_BY_15);

    //Display open transactions
}
```

BLOCK-CHAIN SIMULATION

Function to display open transactions

To draw the open transactions as a line loop.

```
//Display open transactions
void display_open_transactions()
{
    int oty = y - 300;
    RenderString(0, oty + 125, "Open Transactions (Click on any transaction to view it's contents)", GLUT_BITMAP_8_BY_15);
    int tempu = 0;
    char str[3];
    if (no_of_ot >= 9)
    {
        open_transaction_width = 50;
        open_transaction_height = 100;
        open_transaction_font = GLUT_BITMAP_TIMES_ROMAN_10;
    }
    if (no_of_ot == 0)
    {
        RenderString(0, oty + 100, "No open transactions! Use the menu or press 'g' to add a transaction!", GLUT_BITMAP_8_BY_13, 1, 0, 0);
        return;
    }
    for (int i = 1; i <= no_of_ot; i++)
    {
        if (i <= 9)
        {
            to_string_digit(str, i);
        }
        else
        {
            to_string_digits(str, i);
        }
        draw_line_loop(tempu, oty, open_transaction_width, open_transaction_height, 1, 1, 1);
        RenderString(tempu, oty + (open_transaction_height/2), "Transaction", open_transaction_font);
        RenderString(tempu + open_transaction_width - 10, oty + (open_transaction_height/2), str, open_transaction_font);
        tempu = tempu + open_transaction_width + padding;
    }
    if (no_of_ot <= 9)
    {
        to_string_digit(str, no_of_ot);
    }
    else
    {
        to_string_digits(str, no_of_ot);
    }
    RenderString(0, oty - 25, "No of open transactions: ", GLUT_BITMAP_8_BY_13, 0, 1, 0);
    RenderString(150, oty - 25, str, GLUT_BITMAP_8_BY_13, 0, 1, 0);
}
}
```


BLOCK-CHAIN SIMULATION

Function to display individual block

To show the contents of the clicked block.

```

void display_block()
{
    int x1 = 350;
    int y1 = 500;
    char block[3];
    char previous_block[3];
    char t[3];

    if (clicked_block <= 9)
    {
        to_string_digit(block, clicked_block);
        to_string_digit(previous_block, clicked_block - 1);
    }
    else
    {
        to_string_digits(block, clicked_block);
        if ((clicked_block - 1) == 9)
        {
            to_string_digit(previous_block, clicked_block - 1);
        }
        else
        {
            to_string_digits(previous_block, clicked_block - 1);
        }
    }

    if (transactions[clicked_block] <= 9)
    {
        to_string_digit(t, transactions[clicked_block]);
    }
    else
    {
        to_string_digits(t, transactions[clicked_block]);
    }

    if (clicked_block >= manip_block && manip_block != -2)
    {
        draw_polygon(x1, y1, 300, 300, 0, 0, 0);
    }
    else
    {
        draw_polygon(x1, y1, 300, 300);
    }

    RenderString(400, 500, "Viewing Block", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(500, 500, block, GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(400, 550, "Intro / Blockchain / Block", GLUT_BITMAP_HELVETICA_18);
    RenderString(400, 550, "Press B or E to go back!", GLUT_BITMAP_8_BY_13);
    RenderString(x1 + 35, y1 + 250, "Index: ", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(x1 + 75, y1 + 250, block, GLUT_BITMAP_TIMES_ROMAN_24);

    RenderString(x1 + 25, y1 + 325, "Previous Hash: ", GLUT_BITMAP_TIMES_ROMAN_24);
    if (clicked_block > manip_block && manip_block != -2)
    {
        RenderString(x1 + 125, y1 + 325, "Invalid", GLUT_BITMAP_TIMES_ROMAN_24);
    }
    else
    {
        RenderString(x1 + 125, y1 + 325, previous_block, GLUT_BITMAP_TIMES_ROMAN_24);
    }

    RenderString(x1 + 25, y1 + 300, "Transactions: ", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(x1 + 125, y1 + 300, t, GLUT_BITMAP_TIMES_ROMAN_24);
    if (clicked_block >= manip_block && manip_block != -2)
    {
        RenderString(x1, y1 - 50, "Block seems to be invalid", GLUT_BITMAP_8_BY_13, 1, 0, 0);
    }
    else
    {
        RenderString(x1, y1 - 50, "Press M to manipulate the block", GLUT_BITMAP_8_BY_13);
    }

    RenderString(400, 150, "A SIMULATOR BY", GLUT_BITMAP_9_BY_15);
    RenderString(400, 100, "ARUN R SREMOY 1BY17CS032", GLUT_BITMAP_9_BY_15);
    RenderString(400, 50, "CHARAN KALSHETTY 1BY17CS041", GLUT_BITMAP_9_BY_15);

    display_block_transactions();
}

```

BLOCK-CHAIN SIMULATION

Function for each transaction

To show contents of clicked transaction.

```
void display_open_transaction()
{
    int xl = 250;
    int yl = 500;
    char ot[3];
    draw_line_loop(xl, yl, 300, 300, 0, 1, 0);

    if (what_to_do == 7)
    {
        RenderString(400, 550, "Intro / Blockchain / Block / Transaction", GLUT_BITMAP_HELVETICA_18);
    }
    else
    {
        RenderString(400, 550, "Intro / Blockchain / Open Transaction", GLUT_BITMAP_HELVETICA_18);
    }

    if (clicked_ot <= 9)
    {
        to_string_digit(ot, clicked_ot);
    }
    else
    {
        to_string_digits(ot, clicked_ot);
    }

    RenderString(400, 900, "Viewing Transaction", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(540, 900, ot, GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(400, 885, "Press B or Esc to go back!", GLUT_BITMAP_8_BY_13);
    RenderString(xl + 25, yl + 250, "Transaction:", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 125, yl + 250, ot, GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 25, yl + 225, "Sender: ", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 125, yl + 225, ot, GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 25, yl + 200, "Recipient: ", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 125, yl + 200, ot, GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 25, yl + 175, "Amount: ", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(xl + 125, yl + 175, "10.0", GLUT_BITMAP_TIMES_ROMAN_24);
    RenderString(400, 150, "A SIMULATOR BY", GLUT_BITMAP_9_BY_15);
    RenderString(400, 100, "ARUN R SREMOY 1BY17CS022", GLUT_BITMAP_9_BY_15);
    RenderString(400, 50, "CHARAN KALSHETTY 1BY17CS041", GLUT_BITMAP_9_BY_15);
}
```

CHAPTER 6**INTERPRETATION OF RESULTS**

Every block-chain system can be divided into two parts the block-chain itself and the open transactions. The block-chain always has a block with no transactions/ data in it, also known in block chain jargon as a genesis block. Here we see that the simulator has generated a genesis block and is displayed to the user.



Fig: Genesis Block

There are 7 main operations which user can perform.

Creating a new transaction:

New transactions are always placed under the open transactions and can be mined later. To create a new transaction, press the 'T' key on the keyboard. This will create a new transaction and place it under open transactions as shown.



Fig: Output of pressing T three times

Viewing an open transaction:

Any of the open transactions can be viewed by clicking on the transaction block.



Fig: View of Transaction 3

BLOCK-CHAIN SIMULATION

Mining a Block:

Mining a block involves taking all the open transactions and placing them in a block so as to verify their integrity and legitimacy. Press 'M' key on the keyboard to mine a new block in the block-chain.



Fig: Mining a Block

Here we notice that open transactions are empty now and a new block is generated in the block-chain.

Viewing a Block:

Any of the Blocks can be viewed by clicking on the block in the block-chain.

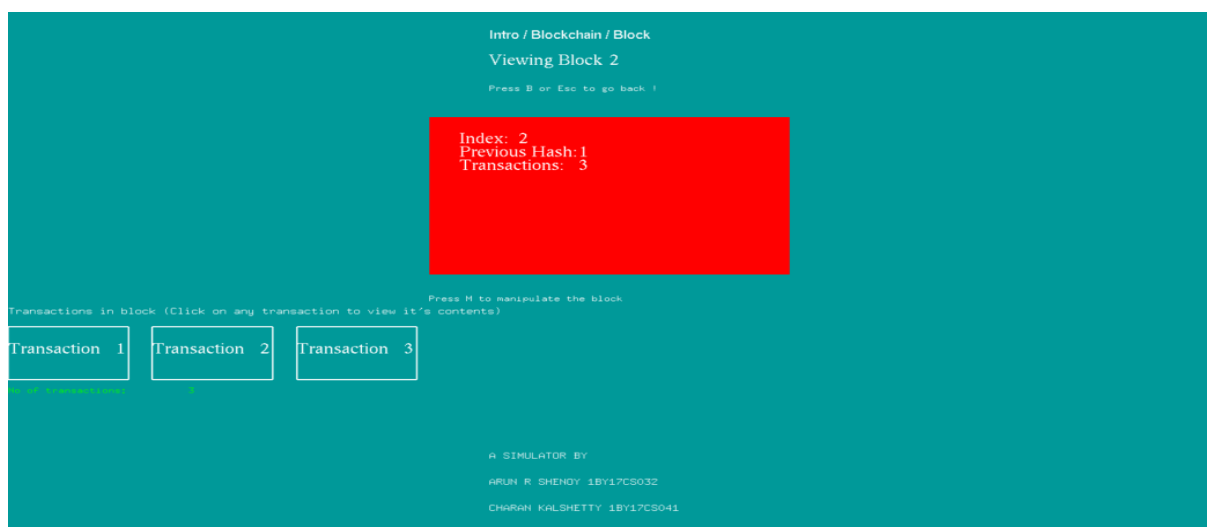


Fig: View of Block 2

BLOCK-CHAIN SIMULATION

Here we notice that the previous three open transactions were verified and placed in the block.



Fig: View of Transaction 2 inside Block 2

Manipulating a Block:



Fig: Manipulated Block

When viewing a particular block in the block-chain the user can press 'M' to manipulate the block and view the effect of the manipulation on the block-chain.

BLOCK-CHAIN SIMULATION



Fig: Effect of manipulating block 2 on the block-chain

Replacing the Block-chain:

The user will not be able to perform operations on a manipulated block-chain and hence must get it replaced by contacting peer nodes. To do this the user must press they key 'C' to simulate contact with a peer node and replace it's blocks.



Fig: Replaced block 2 with it's original

BLOCK-CHAIN SIMULATION

Now the user can continue performing operations on the block-chain.

Resetting the simulation:

In case the user wants to reset the entire simulation he can press the key R.



Fig: Reset Simulation

Alternative way to perform the above operations:

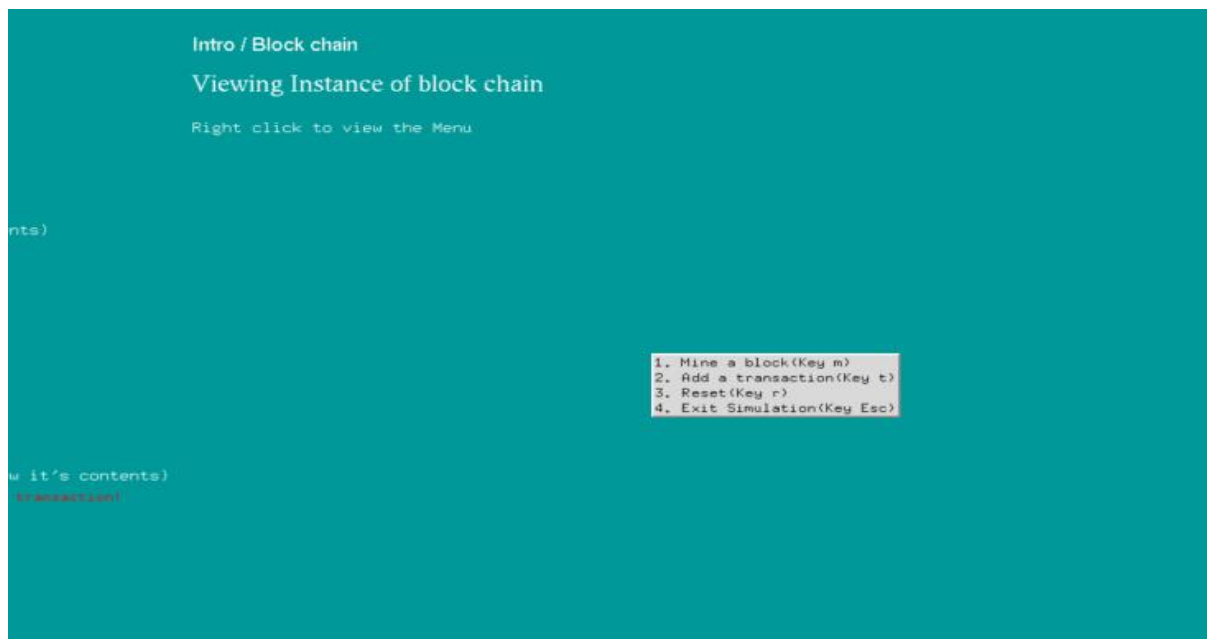


Fig: A zoomed in view of the right click menu

BLOCK-CHAIN SIMULATION

We can also use the right-click menu to perform the above operations.

SOME OTHER FEATURES

Auto-resize:

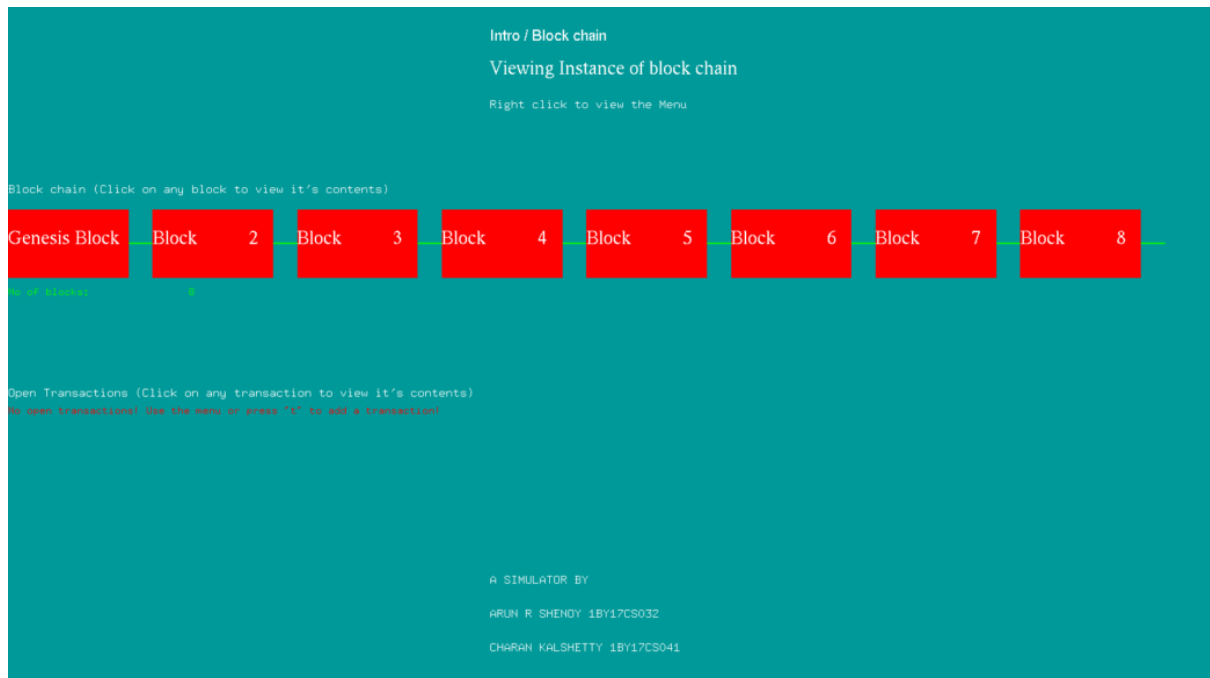


Fig: Block-chain with 8 blocks



Fig: When the 9th block is added

Single key Exit:

We can use the key Esc to exit the simulation at any time. Pressing Esc gives us a confirmation message as below

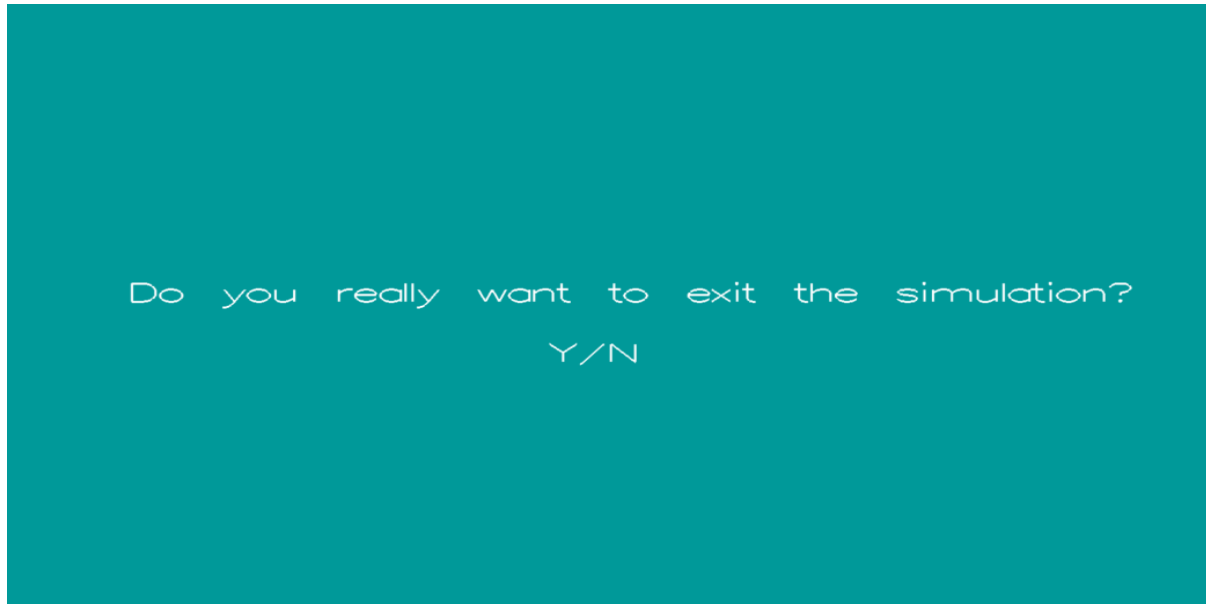


Fig: Exit confirmation message

Breadcrumb:

To help user keep track of where he is during the simulation there is a bread crumb trail on the top that shows the depth in the hierarchy that he is viewing. As you can notice, the trail changes for each image.



Fig: Breadcrumb on top

CONCLUSION:

With the help of computer graphics we have created a Block-chain simulation process. We have created a graphical user interface which is the simulator and the operations performed by the interface are the various block-chain process or the simulation.

FUTUTRE ENHANCEMENTS:

Many future enhancements can be made.

1. Model the entire peer node network.
2. Include more complex data other than transactions.
3. Provide a view of Merkle tree.

BIBLIOGRAPHY

1. Prescribed VTU CG textbook- James D Foley Computer graphics with OpenGL:
Pearson Education
2. <https://www.blockchain.com/> - Block-chain website
3. <https://en.wikipedia.org/> - Wikipedia
4. <https://shankarrajabopal.github.io/>