Programming Club Project Member

PC-01 Multiplayer Tank Royale

Name : Prakash Kumar Jha	Roll Number : ed22b059
Phone No: 6382875760	CGPA: 7.9
Email: ed22b059@smail.iitm.ac.in	Branch: Engineering Design

Task 1: OOPS In C++

Learnings:

- Implementation of OOPs in c++ and creating our own dynamic array data structure,
- Use of template <>

References Used:

a. Geeks for Geeks : Templates, Dynamic Memory Allocation, OOPs in c++
b. CWH (code with Harry)

GitHub Link for code: https://github.com/ProPrak01/PC01_ed22b059_code/tree/main/task1

Task 2: Understanding Multithreading

Brief definition:

Processes: Process is basically a program which is executing (i.e when the computer performs the instructions of program) , isolated from other processes and having their own memory address.

Processes have one or more threads (with a main thread) , and they can be single or multiple threaded , so processes are the superset of its threads.

Threads: They can be termed as the smallest amount of information that lets the cpu executes instructions , there may be multiple threads in a process , and those threads of the same process can having shared memory.

Brief explanation and understanding through an example -

So, if i take an example of opening a notepad when it is run then the operating system creates a new process (so now this notepad process is running with its own memory) and some threads with a main thread.

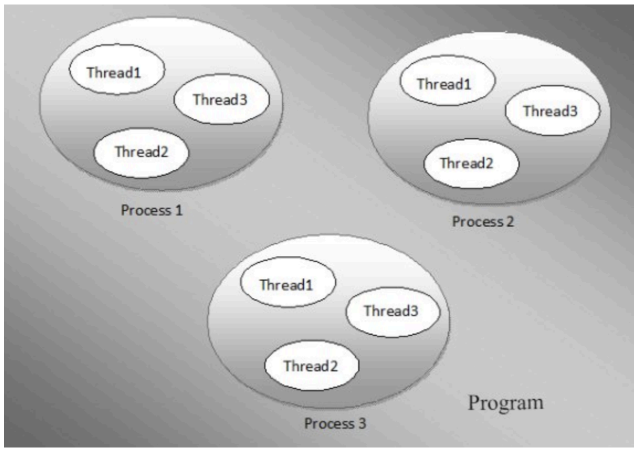
So if you do some 2-3 tasks in notepad which requires to be executes in parallel like you are editing in the file and the file is autosaving in background , so both will be running in different threads.

Post interview:

What is shared between threads ? can a variable declared in main( ) function be accessed by other threads. ?

So,

If a variable is declared within the main function (or any other function), it is scoped to that function and cannot be directly accessed by other threads. Each thread has its own stack, which contains local variables, function call history, and other execution context specific to that thread. These stacks are isolated from each other, and a variable declared in one thread's stack cannot be directly accessed by another thread.



Key Differences:

Thread	Processes
Process will generally consume more resource than a thread	Thread will consume less
Process are generally isolated , i.e each process will have its own isolated memory But still processes can communicate with each other but slower than thread	Threads of same process generally share memory addresses.

GPU:

So , on the basis of architecture the CPU has typically small number of cores. In which each core is highly optimized for sequential execution.

Whereas in GPU contains a very large number of cores . Which are simple and optimized for parallel processing.

Here GPU is based on SIMD architecture , i.e Single Instruction Multiple Data (means same operation can be performed with multiple data simultaneously).

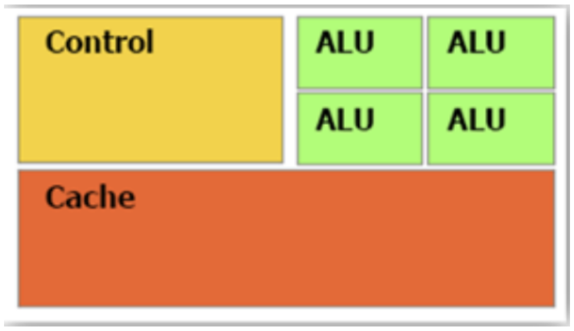
//Understanding SIMD:

A SIMD stream architecture has a single control processor and instruction memory, so only one instruction can be run at any given point in time. That single instruction is copied and ran across each core at the same time. Thus decreasing the time required in case of GPU.

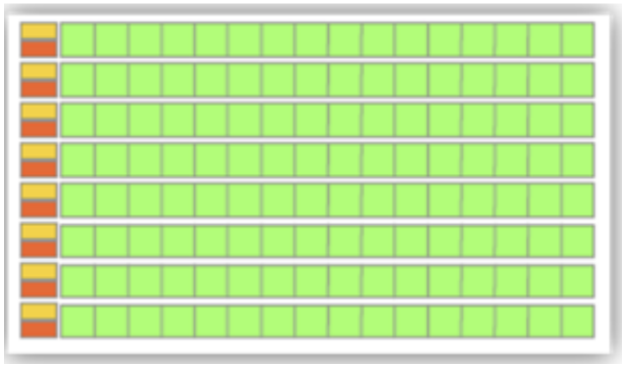
So, By example :

If i want to run a very heavy sequential code with a big data then CPU can out perform the GPU , but if that code has to run for millions of different data sets then the total time taken by GPU will be lesser than the CPU.

Example 2:



CPU architecture



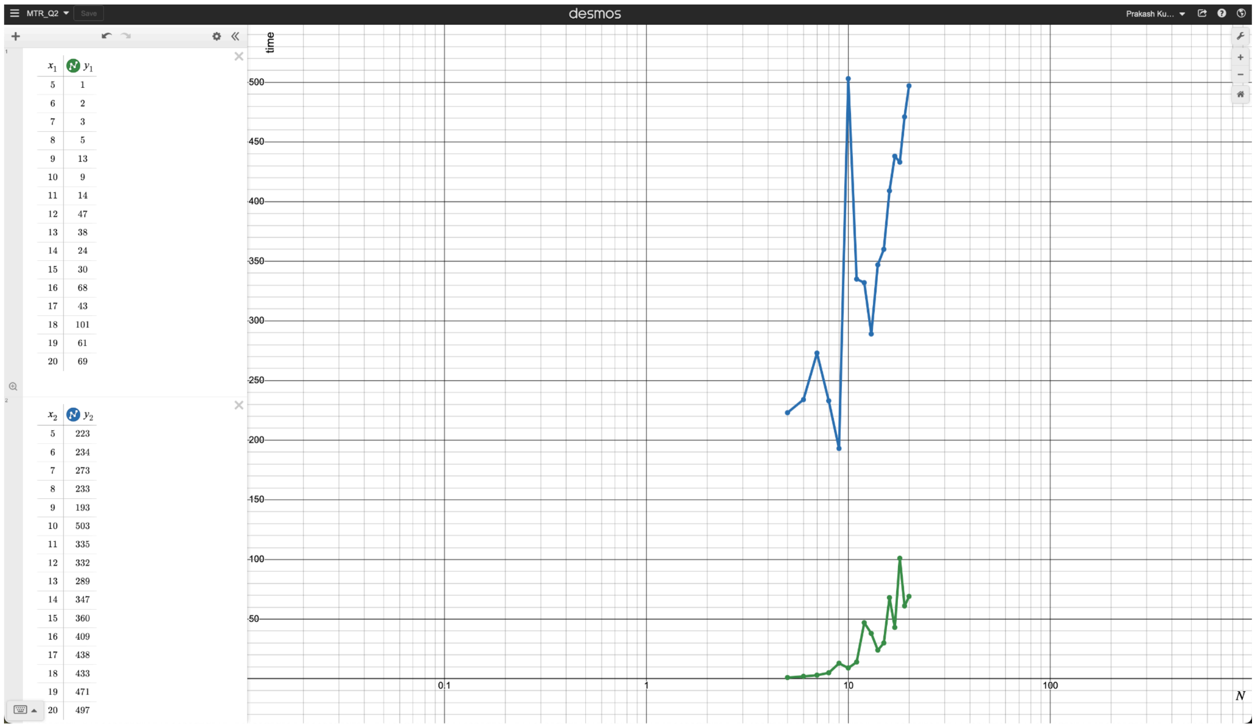
GPU architecture

Matrix multiplication :

https://github.com/ProPrak01/PC01_ed22b059_code/tree/main/task2

Code execution time analysis:

1. For lower values of N (5-20):



x_1	y_1
5	1
6	2
7	3
8	5
9	13
10	9
11	14
12	47
13	38
14	24
15	30
16	68
17	43
18	101
19	61
20	69

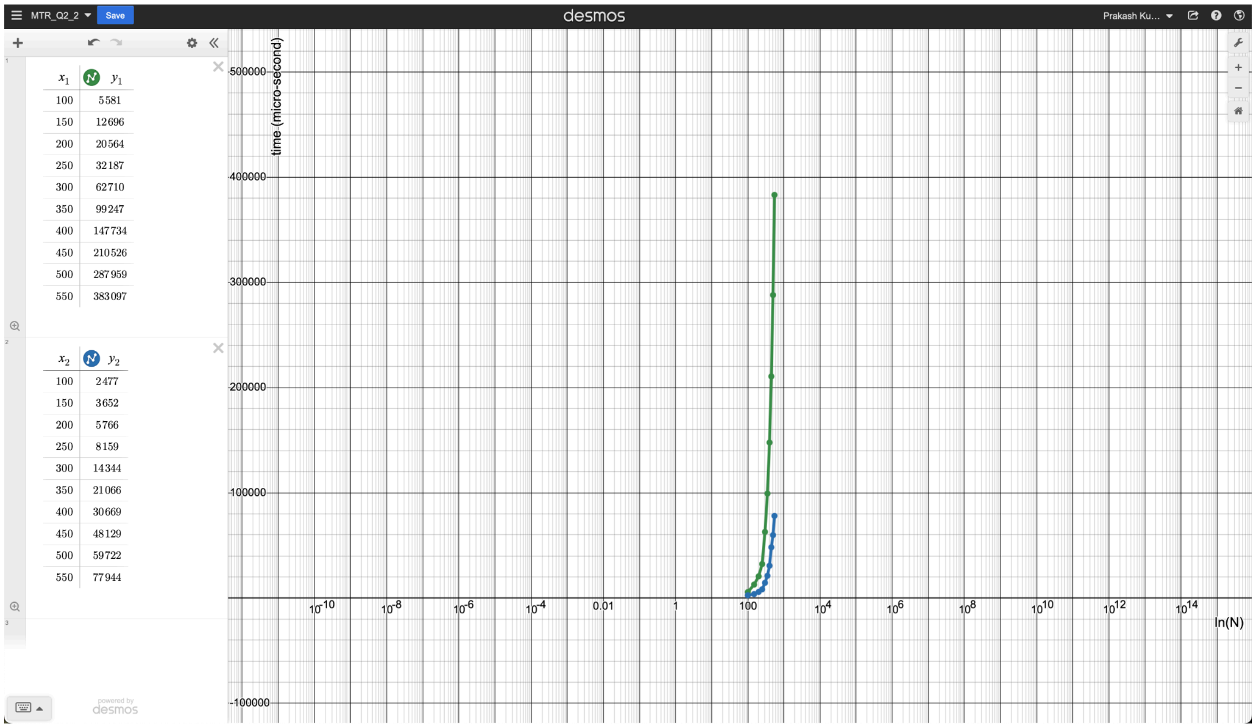
x_2	y_2
5	223
6	234
7	273
8	233
9	193
10	503
11	335
12	332
13	289
14	347
15	360
16	409
17	438
18	433
19	471
20	497

Y-axis: time (micro-seconds)   X-axis: N (row/col number)

green indicates sequential code , data sets : (x1 , y1)
blue indicates parallel thread code, data sets : (x2 , y2)

Conclusion: For lower values of N the sequential code performs better than parallelised code

2. For higher values of N (100-550):



x_1	y_1
100	5581
150	12696
200	20564
250	32187
300	62710
350	99247
400	147734
450	210526
500	287959
550	383097

x_2	y_2
100	2477
150	3652
200	5766
250	8159
300	14344
350	21066
400	30669
450	48129
500	59722
550	77944

Y-axis: time (micro-seconds)   X-axis: ln(N (row/col number))

green indicates sequential code , data sets : (x1 , y1)
blue indicates parallel thread code, data sets : (x2 , y2)

Conclusion: For higher values of N the parallelised code performs better than sequential code

References:

https://eng.libretexts.org/Courses/Delta_College/Operating_System%3A_The_Basics/04%3A_Threads/4.1%3A_Process_and_Threads

https://www.quora.com/What-is-the-difference-between-a-process-and-a-thread

FANG Interview Question | Process vs Thread

https://www.youtube.com/watch?v=_cyVDoyI6NE

CPU vs GPU (What's the Difference?) - Computerphile

https://embeddedcomputing.com/technology/processing/understand-the-mobile-graphics-processing-unit

Question 4.

Reference:

How to create and join threads in C (pthreads). playlist

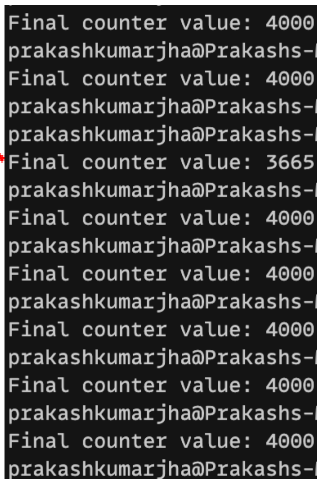
So, if i run the code with argv[1] as (x) some integer

```
pthread_create(&thread1, NULL, increment_counter, &num_increments);
pthread_create(&thread2, NULL, increment_counter, &num_increments);
```

So, due to two threads running parallely increments the counter variable (both the threads are sharing the same "counter" variable) so the expected output should be (2*x)

So when i took x as 1000, so in many runs the answer came out to

Be 4000, but in some cases it showed different value like here



```
Final counter value: 4000
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
prakashkumarjha@Prakashs-
Final counter value: 3665
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
Final counter value: 4000
prakashkumarjha@Prakashs-
```

Reason:

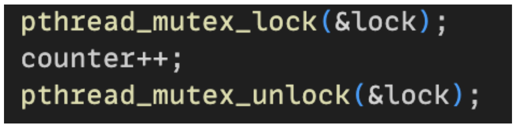
This is an example of "Race condition" occurs when two or more thread

Access shared data and edit it simultaneously.

So for understanding this and correcting this we have to look at how the

Behaviour of thread is , so threads works a check-then-act , where it first checks that what is the the value of that shared data then it acts on the data. But if another thread comes changes the value of that shared data in between then this kind of misbehavior happens.

For correcting this we use 'mutex' a mechanism by which it first locks the counter++ so that the variable is only accessible to one thread at a time



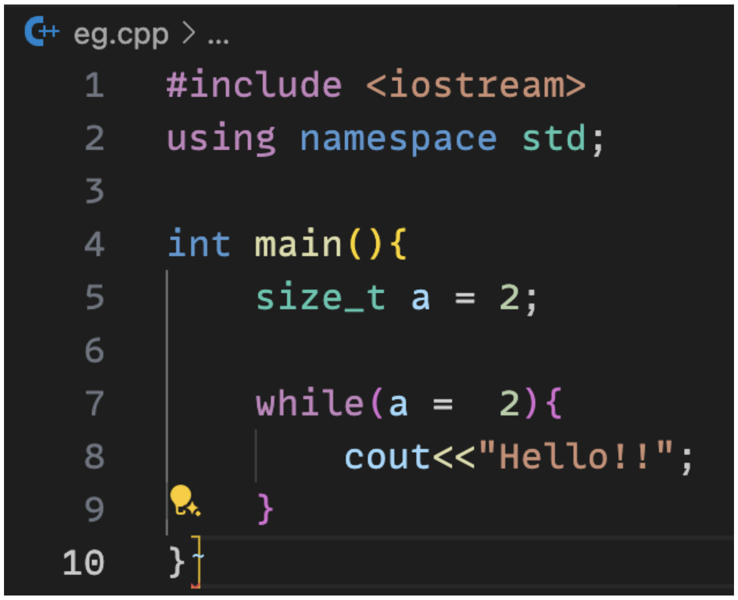
```
pthread_mutex_lock(&lock);
counter++;
pthread_mutex_unlock(&lock);
```

Reference: https://www.geeksforgeeks.org/mutex-lock-for-linux-thread-synchronization/

Volatile: So , if a variable is volatile

It then basically means the compiler will not optimize the code accordingly,

We can understand through an example->



```
#include <iostream>
using namespace std;

int main(){
    size_t a = 2;

    while(a =  2){
        cout<<"Hello!!";
    }
}
```

If i write a code like this, then the compiler may optimize this code to something like while(true),

Because it knows the program is not attempting to change the value of a. So it can put while(true) here by which it does not have to again and again fetch values of a then check the condition.

So to stop the optimization we can use “volatile keyword for variable” so the compiler can know that the value can change from different thread or different devices … or any external way.

In the code give in question 4 we used volatile keyword to ensure the compiler would not optimize the code as different threads are changing the same variable.

Task 3: Understanding Networks and Socket Programming

Sockets: So in basic terms:

Reference involved:https://www.cs.dartmouth.edu/~campbell/cs50/socketprogramming.html

A socket is a way for a program to talk to other programs or devices, either on the same computer or over a network. It's more like a two-way pipe: data you put in one end is sent to the other end, and vice versa.

Or we can say the socket is an endpoint in a network communication system , as they provide an interface for the connection. They are generally used in connection of a client and server .

Most common type of sockets used is the TCP Socket:

They enable error free connections (commonly used in Web development) in this

Common use case is the client can send HTTP requests to the server , and the server can be connected to a database. And CRUD operation can be performed.

Other examples include - WebSockets , UDP Sockets.

fundamental steps involved in creating a socket:

So we will be creating a TCP socket 👍

The steps involve:

Server side

1	Creating a socket using Socket()
2	Binding the socket with a address using bind()
3	Listening for the connection using listen()
4	Accept connection from client with the accept()
5	Sending and receiving data using send() and receive()

Client side

1	Creating a socket using Socket()

2	Connect the socket to the address of the server using the connect()
3	Sending and receiving data using send() and receive()

Question 3 : https://github.com/ProPrak01/PC01_ed22b059_code/tree/main/task3

Reference: https://www.geeksforgeeks.org/socket-programming-in-cpp/

https://stackoverflow.com/questions/3509011/socket-programming-in-c (by user6379829)

Chat gpt reference: https://chatgpt.com/share/0e5b5a45-e427-4847-ba79-163c820f5efc

Task 4: Game Design

Q1

I have some experience in gameDev so , some features which i can suggest:

1. Implementing a single player feature.
2. Multiple tanks (clients) can battle in a single ground.
3. Minimap for client (consider long maps)
4. Scoring System:
    a. Till the end of the match the final tank standing will be winner and others will be ranked accordingly.
    b. Can have ranks (levels) of tanks , by registering them to the server and counting by the number of match played , kills, etc..
5. Powerups :
    a. Ability of different shootings styles depending on the powerup taken.
    b. Increasing health.
    c. Teleportation
    d. Speed changing powerups
    e. Etc..
6. Personalized homepage for every client (for putting server id , stats , leaderboard)

Q2

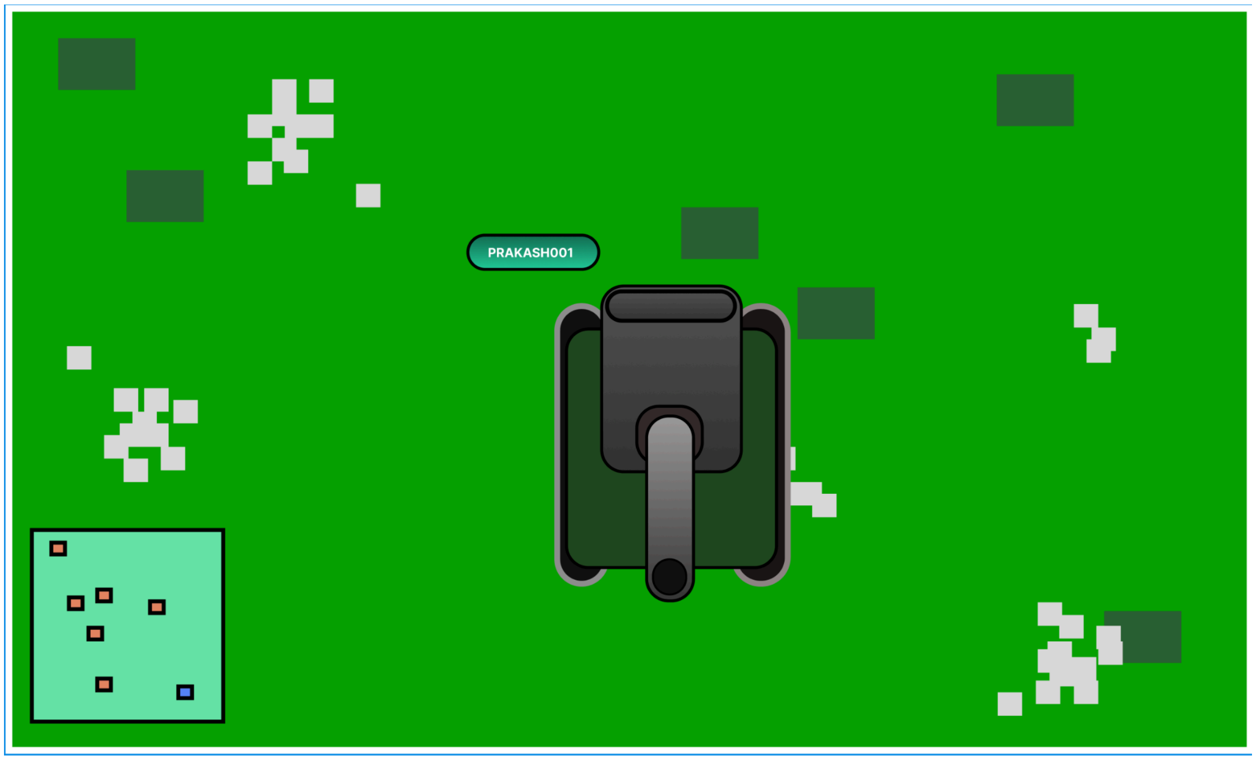
Best is we can use OpenGL for game Interface development.

- Large userbase
- Tutorials available
- Extra libraries: GLEW , GLM

Q3

Mip-Map Design: New Design in figma link

https://www.figma.com/design/urjw7ihoTKqolqy3jlZN8a/Untitled?node-id=0-1&t=j5dkZHfEE4cOQkee-1



Section : About You

1. My motivation to apply for this project stems from my passion for game development. Have worked in parallel programming in rust (in FEM software development Prof. Srikanth Vedantam)

Strengths Include: Have done prior project in envisage Game Dev (created a multiplayer game using Unity ,c#, Netcode)(presented in research conclave 2023)

Worked as Project member in WebOps blockchain club, (in CV Dental project includes Websockets (socket.io) , other Web dev. stacks).(presented in OpenHouse 2024)

Working under Prof. Srikanth Vedantam IITM in (Rust (GUI) and c++ CGAL lib) for computational geometry in developing the FEM software.

Also i am a dual degree student so my intern resume submission is next year (so i am able to devote required time for this).

2. Some of my commitments:
   - Super coord insti webops team .
   - Prof project - as specified above
   - DSA Practice

These commitment are work from home so i can easily manage time for this.