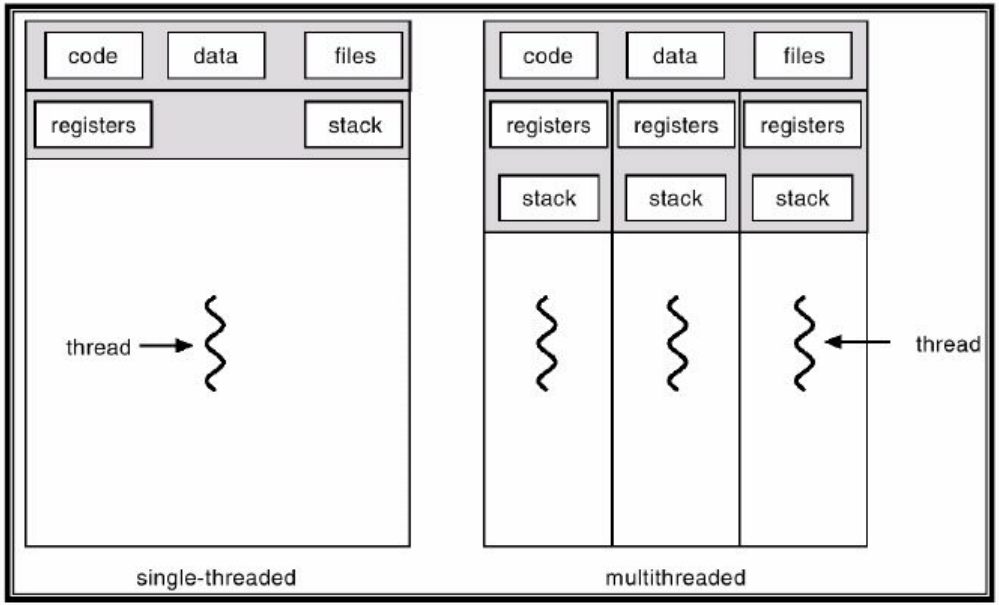
Lab Session 1: Threads

Definition



In operating systems, threads are units of code that are a sequence of instructions that run concurrently to other tasks, sharing memory and resources.

Multithreading vs. multitasking?

https://www.geeksforgeeks.org/difference-between-multiprocessing-and-multithreading/

... in Videogames

- Game Logic?
- AI?
- Networking?
- Art?
- ...

In Unity...

Coroutines vs. Threads

Coroutine: Run bits of each sequence of instructions everytime (simulate parallel execution)

Threads: Run two sequences of instructions in parallel on different threads

When can this go wrong?

2 Threads modifying the same object

One thread modifying and another reading

In what frame are you when you check the variables? Does it matter?

Solution:

Semaphores (or Mutex, or Locks)

Threads in unity

Using:

```
using System.Threading;
using System;
```

Thread myThread = new Thread(myFunction)

```
Thread myThread = new Thread(threadInfiniteSecondPrinter);
myThread.Start();
```

myFunction should return void (example):

```
void threadInfiniteSecondPrinter()
{
    Debug.LogWarning("Starting Thread!");
    System.DateTime myTime;
    while (!exit)
    {
        myTime=System.DateTime.UtcNow;
        while ((System.DateTime.UtcNow-myTime).Seconds < 5f)
        {
            //Debug.Log(System.DateTime.UtcNow);
        }
        Debug.Log("5s passed!");
        //ChangeIndivivdualColorOfAllParticles();

    }
}
```

https://docs.unity3d.com/2020.1/Documentation/Manual/JobSystemMultithreading.html

Ensure data consistency (Threads)

- You want to avoid changing a variable from 2 different threads at the same time
- You want to avoid changing a variable when it's being read by another thread (specially main)

Lock: wait until the variable is not being used anymore

Example:

```
public object myLock = new object();
```

Update:

```
lock(myLock)
{
    transform.position = pos;
}
```

Thread:

```
Vector3 speed = new Vector3(0f,0f,0f);
lock(myLock)
{
    pos=pos+speed;
}
```

Other considerations

- Unity doesn’t like Threads
  - Coroutines
  - Unity API calls

- Thread.IsAlive
- Thread.Abort()
- Thread.Join()

Coroutines in Unity:

Key elements are:

Return an IEnumerator

```
IEnumerator coroutineInfiniteSecondPrinter()
```

yield return XXX (to allow 1 frame update)

XXX can be:

```
while ((System.DateTime.UtcNow-myTime2).Seconds < 10f)
{
    yield return null;
}
```

- null: update one frame and continue execution
- long lasting function: execute one frame and check if the previous function is finished
- Other coroutines

Call with:

```
StartCoroutine("coroutineInfiniteSecondPrinter");
```

https://docs.unity3d.com/Manual/Coroutines.html

Other functions to look for

Yield break

StopCoroutine

StopAllCoroutines

Some Documentation

https://docs.unity3d.com/Manual/Coroutines.html

https://learn.microsoft.com/en-us/dotnet/api/system.threading.thread?view=net-6.0

Advanced concepts:

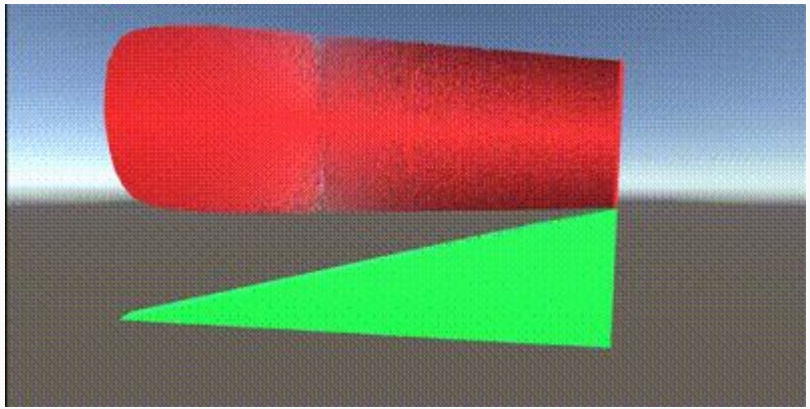
https://www.youtube.com/watch?v=7eKi6NKri6I

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/

HANDOUT

We are going to practice some threading

- In my example I'll be using sorting algorithms
- Realistically, any long function could do, use whatever you see fit.



To Do 0

- Download the P1 handout

What do we have available?

- 1 Array with random values
- 1 empty list to store GOs (for graphical representation purposes)

```
float[] array;
List<GameObject> mainObjects;
public GameObject prefab;

void Start()
{
    mainObjects = new List<GameObject>();
    array = new float[30000];
    for (int i = 0; i < 30000; i++)
    {
        array[i] = (float)Random.Range(0, 1000)/100;
    }
```

To Do 1

- Create a function to print arrays on the console
- Quite trivial, but it will be handy for debug purposes

To do 2

- Time to spawn the geometries, using our array values
- Again, nothing to write home about, just be sure to store the GOs on a list so we can “sort” them later on.

```
Instantiate(prefab, new Vector3((float)i / 1000,
    this.gameObject.GetComponent<Transform>().position.y, 0), Quaternion.identity);
```

To Do 3

“Sorting” the GameObjects list:

- We'll just change the height of every obj in our list to match the values of the array.
- To avoid calling this function once everything is sorted, keep track of new changes to the list.
- If there weren't, you might as well stop calling this function

To Do 4 & 5

- With all our necessary functions set, we can call all 3 of them to spawn the geometries and sort their heights. Depending on how many of them we are instantiating, this can take a while.

- Then, it's time to call the blocking function: BubbleSort(). Do it as a Thread.

```
void bubbleSort()
{
    int i, j;
    int n = array.Length;
    bool swapped;
    for (i = 0; i < n- 1; i++)
    {
        swapped = false;
        for (j = 0; j < n - i - 1; j++)
        {
            if (array[j] > array[j + 1])
            {
                (array[j], array[j+1]) = (array[j+1], array[j]);
                swapped = true;
            }
        }
        if (swapped == false)
            break;
    }
}
```

To Do 6

- Call your ChangeHeights() function to update the GO list when needed.
- Why do we call it in the Main Thread?

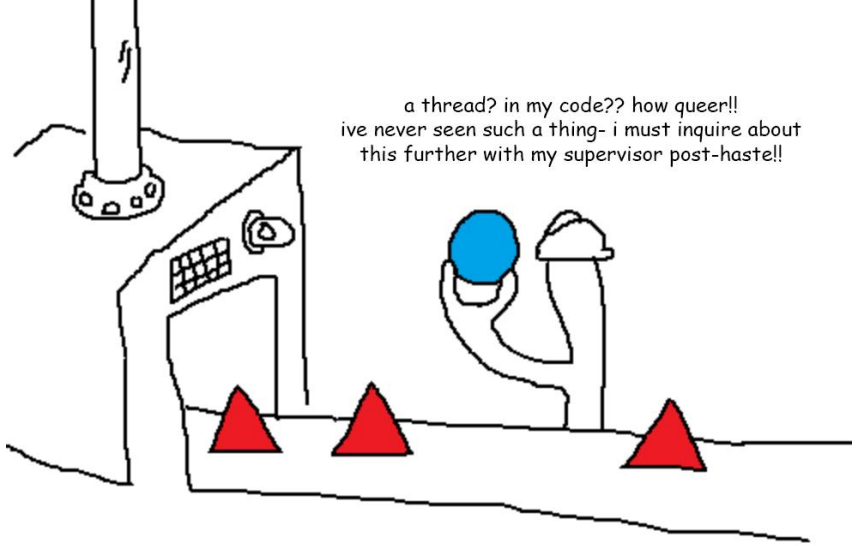
Class Exercise (Deliverable 1)

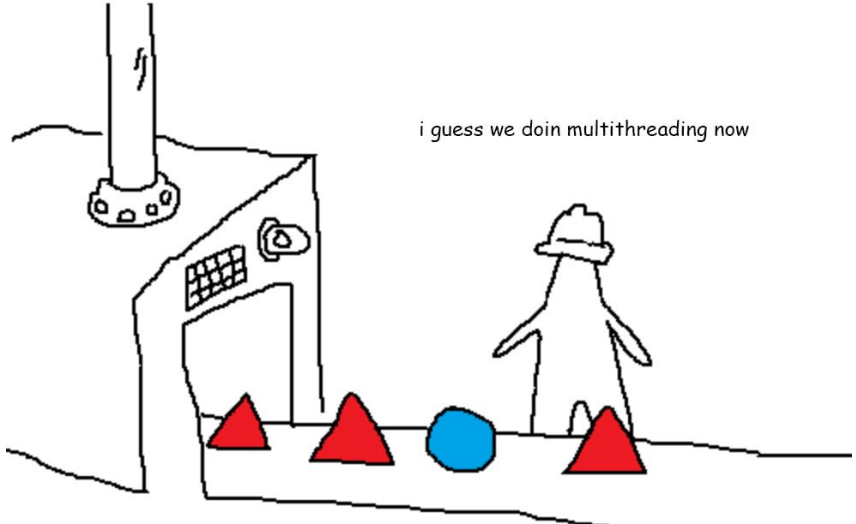
Given what we have seen about threads and concurrent computation:

- Add another sorting algorithm other than Bubble so that we can compare their speed.
- Make sure the frames don’t stop when the sorting starts

If you can finish it in class, show me and I’ll update the grading.

https://www.geeksforgeeks.org/sorting-algorithms/


a thread? in my code?? how queer!!
ive never seen such a thing- i must inquire about
this further with my supervisor post-haste!!


i guess we doin multithreading now