Lab 12 Report - Gipson

# Introduction

Single threaded applications are not necessarily the most efficient. To combat this I will attempt to use multi-threading to improve performance, but not speed.

# Methods

First thing you need to do when threading is create a thread. The constructor parameter is a method name, without parameters. Name the thread something that makes sense, like movementThread for a thread that handles movement. Threads do not play well with shared memory, so whenever possible use variables that are local to the thread. This will require more memory, but will make for more stable programs and games. To start a thread you have to call the Start() method of the thread object you created. You can use variables like IsAlive to tell is a thread is running, and you have the option to use Sleep(int milliseconds) to make a thread pause for the time passed through. Join() can be used to make a thread wait for the a different thread to complete its processing before moving forward. A Lock block can be used to give a thread exclusive rights to a variable. This will make any other thread stop and wait for the locked thread to complete before processing and can caused deadlock however so you must be careful. Something else to watch for is threads not ending. Use Suspend or Abort to stop a thread.

# Conclusion

Threading can be difficult to get right, and can have some very odd results, but its promising. I feel like it would be more useful in event driven programming, or possibly game programming in a different context than what I attempted in this lab. Unity does not support multithreading with it’s API unfortunately, which limits things drastically.

# Post-Lab

1. Explain the what, and why, of what is displayed.
   1. T2, t2. The threads are declared with the variable text to be printed, but sense they are not started until after the text variable has been set to “t2” they will both print out t2, regardless of when the threads were created. The text to print was passed through as a reference to a memory location, not a specific value.

# Code

using UnityEngine;

using System.Collections;

using System.Threading;

/// <summary>

/// Author: Matt Gipson

/// Contact: Deadwynn@gmail.com

/// Domain: www.livingvalkyrie.com

///

/// Description: Lab12ThreadVooDoo

/// </summary>

public class Lab12ThreadVooDoo : MonoBehaviour {

#region Fields

#endregion

void Start() {

Start7();

}

void Start7() {

string text = "t1";

Thread t1 = new Thread(() => print(text));

text = "t2";

Thread t2 = new Thread(() => print(text));

t1.Start();

t2.Start();

}

void Start6() {

Thread worker = new Thread(Go);

print("is worker background? " + worker.IsBackground);

worker.IsBackground = true;

print("is worker background? " + worker.IsBackground);

worker.Start();

}

void Start5() {

Thread.CurrentThread.Name = "main";

Thread worker = new Thread(Go);

worker.Name = "worker";

worker.Start();

Go();

}

void Go() {

print("hello from " + Thread.CurrentThread.Name);

}

void Start4() {

for (int i = 0; i < 10; i++) {

int temp = i;

new Thread(() => print(temp)).Start();

}

}

void Start3() {

for (int i = 0; i < 10; i++) {

new Thread(() => Debug.Log(i)).Start();

}

}

void Start2() {

new Thread(() => {

print("Im running from another thread");

print("this is so cool!");

}).Start();

}

void Start1() {

Thread thread = new Thread(() => DisplayMessage("Hello from the thread!"));

thread.Start();

}

void DisplayMessage(string toPrint) {

print(toPrint);

}

void Update() {}

}

using UnityEngine;

using System.Collections;

using System.Threading;

using JetBrains.Annotations;

/// <summary>

/// Author: Matt Gipson

/// Contact: Deadwynn@gmail.com

/// Domain: www.livingvalkyrie.com

///

/// Description: MovementVooDoo

/// </summary>

public class MovementVooDoo : MonoBehaviour {

#region Fields

public Vector3 moveDirection;

public Vector3 translatingVector = Vector3.left;

public Vector3 currPos;

public float speed;

public float delta;

Thread thread;

#endregion

void Start() {

thread = new Thread(Move);

//initialize thread

//start thread

thread.Start();

//start invoke repeating

InvokeRepeating("ChangeDirection", 5, 5);

}

void Move() {

while (translatingVector != currPos) {

//print("thread running");

translatingVector = Vector3.Lerp(translatingVector, moveDirection, delta \* speed);

}

}

void ChangeDirection() {

print("called");

moveDirection = new Vector3(Random.Range(0,5), Random.Range(0,5), Random.Range(0,5));

}

void Update() {

transform.Translate(translatingVector);

currPos = transform.position;

delta = Time.deltaTime;

}

void OnTriggerEnter(Collider other) {

//print("triggered by " + other.name);

if (other.tag == "Enemy" ) {

//print("in if block");

HealthVooDoo health = GetComponent<HealthVooDoo>();

health.health--;

}

}

}

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

/// <summary>

/// Author: Matt Gipson

/// Contact: Deadwynn@gmail.com

/// Domain: www.livingvalkyrie.com

///

/// Description: HealthVooDoo

/// </summary>

public class HealthVooDoo : MonoBehaviour {

#region Fields

public int health = 5;

public Text healthText;

#endregion

void Update() {

healthText.text = "health: " + health;

}

}