Advanced Game Design

Paul Craven

Apr 11, 2022

RESOURCES:

CHAPTER

ONE

CAMEL CODE REVIEW

Here are examples of the "Camel" game. Our goal here is to do code reviews on this code.

Before the code review, think about:

1. First, list what are the goals of the code review.
2. Look for common mistakes. Keep a to-do list.
    1. Can a person drink more water than is in the canteen?
    2. Do we mis-calculate how far back the people are?
    3. Can the chasing people skip past the person and miss seeing them?
    4. Can we both win and lose the game at the same time? Or otherwise get conflicting messages?
3. Quantify effectiveness of your code review. (Bugs found, changes made, etc.)
4. Code reviews often include work on unit-tests. We aren't doing that here but keep it in mind.
5. Code reviews should be less than 400 lines and 60 minutes.

1.1 Camel Version 1

```
using System;

namespace CamelGame
{
        class Program
        {
                private static void Choices()
                {
                        Console.WriteLine(" ");
                        Console.WriteLine("A. Drink from your canteen.");
                        Console.WriteLine("B. Ahead moderate speed.");
                        Console.WriteLine("C. Ahead full speed.");
                        Console.WriteLine("D. Stop and rest.");
                        Console.WriteLine("E. Status check.");
                        Console.WriteLine("Q. Quit.");
                        Console.WriteLine(" ");
                }
                private static void DistanceTraveled(int camelMovement)
                {
```

(continues on next page)

(continued from previous page)

```
                        Console.WriteLine(" ");
                        Console.WriteLine("You traveled " + camelMovement + " miles.");
                }
                private static void ChoiceA(ref int thirst, ref int drinks)
                {
                        if (drinks > 0)
                        {
                                drinks -= 1;
                                thirst = 0;
                        }
                        else Console.WriteLine("You are out of water.");
                }
                private static void ChoiceB(ref int milesTraveled, ref int thirst, ref␣
↪int camelTiredness, ref int drinks, ref int nativeDistance, Random random)
                {
                        int camelMovement = random.Next(5, 12);
                        int nativeMovement = random.Next(7, 14);
                        milesTraveled += camelMovement;
                        thirst += 1;
                        camelTiredness += 1;
                        nativeDistance += nativeMovement;
                        int oasisFound = random.Next(1, 20);
                        if (oasisFound == 1)
                        {
                                Oasis(out thirst, out camelTiredness, out drinks);
                        }
                        DistanceTraveled(camelMovement);
                }
                private static void ChoiceC(ref int milesTraveled, ref int thirst, ref␣
↪int camelTiredness, ref int drinks, ref int nativeDistance, Random random)
                {
                        int camelMovement = random.Next(10, 20);
                        int nativeMovement = random.Next(7, 14);
                        int addTiredness = random.Next(1, 3);
                        milesTraveled += camelMovement;
                        thirst += 1;
                        camelTiredness += addTiredness;
                        nativeDistance += nativeMovement;
                        int oasisFound = random.Next(1, 20);
                        if (oasisFound == 1)
                        {
                                Oasis(out thirst, out camelTiredness, out drinks);
                        }
                        DistanceTraveled(camelMovement);
                }
                private static void ChoiceD(ref int nativeDistance, ref int␣
↪camelTiredness, Random random)
                {
                        Console.WriteLine("The Camel is happy");
                        int nativeMovement = random.Next(7, 14);
                        nativeDistance += nativeMovement;
                        camelTiredness = 0;
```

(continues on next page)

(continued from previous page)

```
69                  }
70                  private static void ChoiceE(int milesTraveled, int drinks, int␣
↪nativeDistance)
71                  {
72                          Console.WriteLine("Miles traveled: " + milesTraveled);
73                          Console.WriteLine("Drinks in canteen: " + drinks);
74                          Console.WriteLine("The natives are " + (milesTraveled -␣
↪nativeDistance) + " miles behind you.");
75                  }
76                  private static void Oasis(out int thirst, out int camelTiredness, out␣
↪int drinks)
77                  {
78                          Console.WriteLine("You found an oasis!");
79                          Console.WriteLine("The camel is rested and your thirst and␣
↪canteen are replenished.");
80                          thirst = 0;
81                          drinks = 3;
82                          camelTiredness = 0;
83                  }
84                  static void Main(string[] args)
85                  {
86                          int milesTraveled = 0;
87                          int thirst = 0;
88                          int camelTiredness = 0;
89                          int drinks = 3;
90                          int nativeDistance = -20;
91                          string choice;
92                          string playAgain;
93                          bool done = false;
94                          Random random = new Random();
95
96                          Console.WriteLine("Welcome to Camel!");
97                          Console.WriteLine("You have stolen a camel to make your way␣
↪across the great Mobi desert.");
98                          Console.WriteLine("The natives want their camel back and are␣
↪chasing you down! Survive your desert trek and out run the natives.");
99
100                         while (!done)
101                         {
102                                 Choices();
103                                 Console.Write("Enter Choice: ");
104                                 choice = Console.ReadLine();
105                                 if (string.Equals("Q", choice.ToUpper()))
106                                 {
107                                         done = true;
108                                 }
109
110                                 else if (string.Equals("A", choice.ToUpper()))
111                                 {
112                                         ChoiceA(ref thirst, ref drinks);
113                                 }
114
```

(continues on next page)

(continued from previous page)

```
                                else if (string.Equals("B", choice.ToUpper()))
                                {
                                        ChoiceB(ref milesTraveled, ref thirst, ref␣
↪camelTiredness, ref drinks, ref nativeDistance, random);
                                }

                                else if (string.Equals("C", choice.ToUpper()))
                                {
                                        ChoiceC(ref milesTraveled, ref thirst, ref␣
↪camelTiredness, ref drinks, ref nativeDistance, random);
                                }

                                else if (string.Equals("D", choice.ToUpper()))
                                {
                                        ChoiceD(ref nativeDistance, ref camelTiredness,␣
↪random);
                                }

                                else if (string.Equals("E", choice.ToUpper()))
                                {
                                        ChoiceE(milesTraveled, drinks, nativeDistance);
                                }

                                if (thirst > 6)
                                {
                                        Console.WriteLine("You died of thirst.");
                                        done = true;
                                }

                                else if (thirst > 4)
                                {
                                        Console.WriteLine("You are thirsty.");
                                }

                                if (camelTiredness > 8 & !done)
                                {
                                        Console.WriteLine("Your camel is dead.");
                                        done = true;
                                }

                                else if (camelTiredness > 5)
                                {
                                        Console.WriteLine("Your camel is getting tired.
↪");
                                }

                                if (milesTraveled - nativeDistance <= 0 & !done)
                                {
                                        Console.WriteLine("The natives caught you!");
                                        done = true;
                                }

```

(continues on next page)

(continued from previous page)

```
                                else if (milesTraveled - nativeDistance <= 15)
                                {
                                        Console.WriteLine("The natives are getting close!
↪");
                                }

                                if (milesTraveled >= 200 & !done)
                                {
                                        Console.WriteLine("You escaped the natives!");
                                        done = true;
                                }

                                if (done)
                                {
                                        Console.Write("Play Again? (Y/N) ");
                                        playAgain = Console.ReadLine();
                                        if (string.Equals("Y", playAgain.ToUpper()))
                                        {
                                                done = false;
                                        }
                                        else if (string.Equals("N", playAgain.ToUpper()))
                                        {
                                                Console.WriteLine("Thanks for playing!");
                                        }
                                }
                        }
                }
        }
}
```

1.2 Camel Version 2

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace Camel
{
    class Program
    {
        // Initialize variables for use throughout the entire program
        static int playerPosition;
        static int hadesPosition;
        static int gameLength;
        static bool done;
        static int energy;
        static int maxEnergy = 20;
        static int drachmas;
        static int maxShops = 4;
        static int hadesMovement;
```

(continues on next page)

(continued from previous page)

```
        static int turnCounter;
        static int positionDifference;
        static string distanceStatement;
        static string energyStatement;
        static int[] shops;
        static int movementModifier;
        static int hadesMovementModifier;
        static Dictionary<string, int> shopItems;
        static int speedBoostTurns;
        static int maxSpeedBoost = 5;
        static int roadBlockTurns;
        static int maxRoadBlock = 5;
        static bool purchaseMade;
        static bool moved;
        static bool exitShop;
        static string lineBreak = "------------------------------------------------------------
↪--------------------------------";

        static void Main(string[] args)
        {
            // ---------------------------------------- SETUP FUNCTIONS FOR GAME USE ---
↪---------------------------------------
            void InitializeGame()
            {
                // Initialize game variables and general setup.
                hadesPosition = 0;
                // Set player position relative to Hades at the beginning of the game.
                playerPosition = SetPlayerPosition(hadesPosition);
                // Randomize the length that the player must travel.
                SetGameLength();
                // Get our list of shop locations.
                shops = DisperseShops();
                energy = maxEnergy;
                turnCounter = 0;
                // This is used to keep our game running until the player wins or is␣
↪caught by Hades.
                done = false;
                hadesMovementModifier = 0;
                movementModifier = 0;
                shopItems = new Dictionary<string, int>();
                InitializeShop();

                string gameStartTutorial;
                gameStartTutorial = "You are in ancient Greece and have just completed␣
↪an undercover recon mission for Zeus. Hades has discovered " +
                    "what you have done and is now chasing you while you make your way␣
↪back to Olympus! Get back to Olympus before Hades makes you pay " +
                    "the price!";

                Console.WriteLine(gameStartTutorial);
                Console.WriteLine(lineBreak);
            }
```

(continues on next page)

(continued from previous page)

```

            // This function will set the Game's length every time it is started up.
            int SetGameLength()
            {
                gameLength = RandomNumber(45, 60);
                return gameLength;
            }

            // Function for generating random numbers within the game.
            int RandomNumber(int min, int max)
            {
                Random random = new Random();
                return random.Next(min, max);
            }

            // This function sets the players initial starting position with respect to the enemy's position.
            int SetPlayerPosition(int hadesLocation)
            {
                int playerLocation;
                playerLocation = hadesLocation + RandomNumber(1, 6);
                return playerLocation;
            }

            int SetHadesPosition()
            {
                hadesMovement = RandomNumber(2, 5);
                if (hadesMovement - hadesMovementModifier < 0)
                {
                    hadesMovement = 0;
                }
                else
                {
                    hadesMovement = hadesMovement - hadesMovementModifier;
                }
                hadesPosition += hadesMovement;
                return hadesPosition;
            }

            int[] DisperseShops()
            {
                int previousShop;
                int interval;
                int[] shopLocations = new int[maxShops];

                previousShop = 0;

                for (int i = 0; i < maxShops; i++)
                {
                    interval = RandomNumber(4, 10);
                    if (previousShop + 4 < gameLength && (previousShop + interval) < gameLength)
```

(continues on next page)

(continued from previous page)

```
                    {
                        shopLocations[i] = previousShop + interval;
                        previousShop = shopLocations[i];
                    }
                }

                return shopLocations;
            }

            string GetEnergyStatement(int energy)
            {
                if (energy <= 3)
                {
                    return "You are exhausted...";
                }
                else if (energy > 3 && energy <= 10)
                {
                    return "You are starting to get tired...";
                }
                else if (energy > 10 && energy <= 15)
                {
                    return "You still feel moderately energetic.";
                }
                else
                {
                    return "You are pulsing with energy!";
                }
            }

            void CheckForAncientRuins()
            {
                int random;
                int drachmasGained;
                random = RandomNumber(0, 26);
                drachmasGained = RandomNumber(5, 15);
                if (random == 12)
                {
                    Console.WriteLine("You have stumbled upon some ancient ruins. You␣
↪have found " + drachmasGained + " drachmas and your energy has been restored!");
                    drachmas += drachmasGained;
                    energy = maxEnergy;
                }
            }

            void Purchase(string itemName, int price)
            {
                string shopStatement = "";

                if (itemName.Equals("Energy Drink") && drachmas >= price)
                {
                    energy = maxEnergy;
                    shopStatement = "Your energy has been replenished!";
```

(continues on next page)

(continued from previous page)

```
                    shopItems.Remove("Energy Drink");
                    drachmas -= price;
                    purchaseMade = true;
                }
                else if (itemName.Equals("Speed Boost") && drachmas >= price)
                {
                    movementModifier = RandomNumber(2, 4);
                    speedBoostTurns = maxSpeedBoost;

                    shopStatement = "Your movement speed has been boosted by " +␣
↪movementModifier +
                        " for " + speedBoostTurns + " turns!";
                    shopItems.Remove("Speed Boost");
                    drachmas -= price;
                    purchaseMade = true;
                }
                else if (itemName.Equals("Road Block") && drachmas >= price)
                {
                    hadesMovementModifier = RandomNumber(1, 3);
                    roadBlockTurns = maxRoadBlock;
                    shopStatement = "You have injured Hades and have restricted his␣
↪movement by " +
                        hadesMovementModifier + " for " + roadBlockTurns + " turns!";
                    shopItems.Remove("Road Block");
                    drachmas -= price;
                    purchaseMade = true;
                }
                else if (drachmas < price)
                {
                    Console.WriteLine("You cannot afford that!");
                    purchaseMade = false;
                }
                else
                {
                    Console.WriteLine("You have decided to leave.");
                    exitShop = true;
                }
                if (purchaseMade)
                {
                    Console.WriteLine("You have purchased one " + itemName);
                    Console.WriteLine(shopStatement);
                }
            }

            void OpenShop()
            {
                string shopStatement = "You have stumbled upon a traveling merchant!";
                int i = 1;
                Console.WriteLine(shopStatement);
                Dictionary<int, string> itemList = new Dictionary<int, string>();

                foreach (KeyValuePair<string, int> item in shopItems)
```

(continues on next page)

(continued from previous page)

```
                {
                    string displayItem = i + ". " + item.Key + " -------- " + item.Value;
                    Console.WriteLine(displayItem);
                    itemList.Add(i, item.Key);
                    i += 1;
                }

                string leave = i + ". Leave";
                itemList.Add(i, "Leave");
                string playerDrachmas = "Your current held drachmas: " + drachmas;
                string buy = "Would you like to make a purchase?";
                Console.WriteLine(leave);
                Console.WriteLine(playerDrachmas);
                Console.WriteLine(buy);

                purchaseMade = false;
                exitShop = false;

                while (!purchaseMade && !exitShop)
                {
                    string userInput = Console.ReadLine();

                    try
                    {
                        string itemName;
                        int price;

                        itemList.TryGetValue(Convert.ToInt32(userInput), out itemName);
                        shopItems.TryGetValue(itemName, out price);
                        Purchase(itemName, price);
                        exitShop = true;
                    }
                    catch
                    {
                        Console.WriteLine("Please enter a valid number.");
                    }
                }
            }

            void InitializeShop()
            {
                shopItems.Add("Energy Drink", 10);
                shopItems.Add("Speed Boost", 25);
                shopItems.Add("Road Block", 15);
            }

            void IncrementItemDuration()
            {
                if (speedBoostTurns > 0)
                {
                    speedBoostTurns -= 1;
                }
```

(continues on next page)

(continued from previous page)

```
                if (roadBlockTurns > 0)
                {
                    roadBlockTurns -= 1;
                }
            }

            void GetUserDecision()
            {
                string slow;
                string medium;
                string fast;
                string rest;
                string input;
                string status;
                string quit;
                bool decided;
                int drachmasGained = 0;

                decided = false;

                slow = "1. Slow and Steady...";
                medium = "2. Keep a moderate pace.";
                fast = "3. Full steam ahead!!!";
                rest = "4. Stop and take a rest...";
                status = "5. Journey Status.";
                quit = "6. Quit Game.";

                if (positionDifference >= 10)
                {
                    Console.WriteLine("Hades is very far away...");
                }
                else if (positionDifference >= 6)
                {
                    Console.WriteLine("Hades is getting closer.");
                }
                else
                {
                    Console.WriteLine("Hades is right on your tail!!!");
                }
                Console.WriteLine();
                Console.WriteLine(slow + "\n" + medium + "\n" + fast + "\n" + rest + "\n
↪" + status + "\n" + quit);
                Console.WriteLine("What would you like to do?");

                while (!decided)
                {
                    // Get user input and set up if statements to evaluate user input.
                    input = Console.ReadLine();

                    int distanceTraveled;
                    int energyUsed;

```

(continues on next page)

(continued from previous page)

```
                    // If player has decided to take it slow...
                    if (input.Equals("1"))
                    {
                        distanceTraveled = (RandomNumber(1, 3) + movementModifier);
                        energyUsed = RandomNumber(-3, -1);
                        playerPosition += distanceTraveled;
                        distanceStatement = "You have decided to play it safe, and have␣
↪traveled " + distanceTraveled + " miles. ";
                        energyStatement = GetEnergyStatement(energy);
                        drachmasGained = RandomNumber(1, 3);

                        // Make sure we are not surpassing the maximum energy cap.
                        if (energy - energyUsed < maxEnergy - energyUsed)
                        {
                            energy -= energyUsed;
                        }
                        else
                        {
                            energy = maxEnergy;
                        }

                        IncrementItemDuration();

                        decided = true;
                    }
                    // If player has decided on medium travel speed...
                    else if (input.Equals("2") && energy >= 4)
                    {
                        distanceTraveled = (RandomNumber(3, 5) + movementModifier);
                        energyUsed = RandomNumber(2, 4);
                        playerPosition += distanceTraveled;
                        energy -= energyUsed;
                        distanceStatement = "You have decided to move at a steady pace,␣
↪and have traveled " + distanceTraveled + " miles. ";
                        energyStatement = GetEnergyStatement(energy);
                        drachmasGained = RandomNumber(2, 4);
                        drachmas += drachmasGained;
                        IncrementItemDuration();

                        decided = true;
                    }
                    // If player has decided to go full speed...
                    else if (input.Equals("3") && energy >= 8)
                    {
                        distanceTraveled = (RandomNumber(4, 8) + movementModifier);
                        energyUsed = RandomNumber(6, 8);
                        drachmasGained =
                        playerPosition += distanceTraveled;
                        energy -= energyUsed;
                        distanceStatement = "You have decided to travel at full speed,␣
↪and have traveled " + distanceTraveled + " miles. ";
                        energyStatement = GetEnergyStatement(energy);
```

(continues on next page)

(continued from previous page)

```
                        drachmasGained = RandomNumber(3, 6);
                        IncrementItemDuration();

                        decided = true;
                    }
                    // If player has decided to rest...
                    else if (input.Equals("4"))
                    {
                        energyUsed = RandomNumber(-10, -6);
                        energy -= energyUsed;
                        distanceStatement = "You have decided to take a rest and recover your energy. ";
                        energyStatement = GetEnergyStatement(energy);
                        IncrementItemDuration();

                        decided = true;
                    }
                    else if (input.Equals("5"))
                    {
                        Console.WriteLine("----------------- STATUS REPORT --------------\nEnergy: " + GetEnergyStatement(energy) +
                            "\nDrachmas: " + drachmas + "\nHades is " + positionDifference + " miles behind you.");
                        decided = false;
                    }
                    else if (input.Equals("6"))
                    {
                        bool decisionMade = false;
                        Console.WriteLine("Are you sure you would like to quit?\n1. Yes\n2. No");
                        string choice;

                        while (!decisionMade)
                        {
                            choice = Console.ReadLine();

                            if (choice.Equals("1"))
                            {
                                Console.WriteLine("You have exited the game.");
                                done = true;
                                decided = true;
                                decisionMade = true;
                            }
                            else if (choice.Equals("2"))
                            {
                                decisionMade = true;
                            }
                            else
                            {
                                Console.WriteLine("Please enter either 1 or 2.");
                            }
                        }
```

(continues on next page)

(continued from previous page)

```
                    }
                    // If player has entered anything that is not above or does not have enough energy...
                    else
                    {
                        if ((input.Equals("2") && energy < 4) || (input.Equals("3") && energy < 8))
                        {
                            Console.WriteLine("You do not have enough energy for that!");
                        }
                        else
                        {
                            Console.WriteLine("That is not an option, please enter a number between 1 and 4.");
                        }
                        // Keep loop running.
                        decided = false;
                    }
                }

                // Update Hades position and print out the game status.
                if (!done)
                {
                    SetHadesPosition();
                    Console.WriteLine(distanceStatement + energyStatement);
                    Console.WriteLine("While traveling you found " + drachmasGained + " drachmas!");
                    drachmas += drachmasGained;
                    turnCounter += 1;
                }

                for (int i = 0; i < shops.Length; i++)
                {
                    if (playerPosition == shops[i])
                    {
                        OpenShop();
                    }
                }
            }
            // ------------------------------------------------------------------------------------------

            // ----------------------------------------------------MAIN GAME SETUP-------------------------------------------
            InitializeGame();

            // Run this loop while the game is not over.
            while (!done)
            {
                string input;
                bool playAgainDecision;
                if (playerPosition < gameLength)
```

(continues on next page)

(continued from previous page)

```
                {
                    if (playerPosition <= hadesPosition)
                    {
                        Console.WriteLine("You were caught! Game Over!");
                        playAgainDecision = false;
                        Console.WriteLine("Would you like to try again?\n1. Yes\n2. No");

                        while (!playAgainDecision)
                        {
                            input = Console.ReadLine();

                            if (input.Equals("1"))
                            {
                                InitializeGame();
                                playAgainDecision = true;
                                done = false;
                            }
                            else if (input.Equals("2"))
                            {
                                playAgainDecision = true;
                                done = true;
                            }
                            else
                            {
                                Console.WriteLine("Invalid Input. Please enter either 1␣
↪or 2.");

                                playAgainDecision = false;
                            }
                        }
                    }
                    else
                    {
                        CheckForAncientRuins();
                        GetUserDecision();
                        positionDifference = playerPosition - hadesPosition;
                        Console.WriteLine(lineBreak);
                    }
                }
                else
                {
                    Console.WriteLine("You win!");
                    playAgainDecision = false;
                    while (!playAgainDecision)
                    {
                        input = Console.ReadLine();

                        if (input.Equals("1"))
                        {
                            InitializeGame();
                            playAgainDecision = true;
                            done = false;
                        }
```

(continues on next page)

(continued from previous page)

```
                        else if (input.Equals("2"))
                        {
                            playAgainDecision = true;
                            done = true;
                        }
                        else
                        {
                            Console.WriteLine("Invalid Input. Please enter either 1 or 2.
↪");
                            playAgainDecision = false;
                        }
                    }
                }
            }
        }
        // ------------------------------------------------------------------------------
↪----------------------------------------
    }
}
```

1.3 Camel Version 3

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CamelGame
{
    class Program
    {
        private static readonly int MILES_TO_HIDEOUT = 200;

        private static bool done;
        private static bool win;
        private static bool quit;
        private static int milesTraveled;
        private static int fillupsLeft;
        private static int policeMilesTraveled;
        private static int gasTankLeft;
        private static char userInput;
        private static bool validInput;
        private static readonly Random rand = new Random();

        static bool FoundOasis(int findingNumber)
        {
            if (findingNumber == 15)
            {
                return true;
```

(continues on next page)

(continued from previous page)

```
            }
            else
            {
                return false;
            }
        }

        static void Main()
        {
            bool playAgain = true;

            while (playAgain)
            {
                Console.WriteLine("Welcome to Bank Heist!\n" +
                "You have stolen one-million dollars from a bank and must escape to your secret hide out.\n" +
                "The police are hot on your tail and will stop at nothing to catch you!\n" +
                "Out run the cops and escape to your hideout to keep your freedom.\n");

                done = false;
                win = false;
                validInput = true;

                milesTraveled = 0;
                gasTankLeft = 0;
                fillupsLeft = 3;

                policeMilesTraveled = -20;


                while (!done)
                {
                    Console.WriteLine();
                    Console.WriteLine("A. Ahead moderate speed.\n" +
                        "B. Ahead full speed.\n" +
                        "C. Stop to fill up the gas tank.\n" +
                        "D. Status check.\n" +
                        "Q. Quit.\n");

                    Console.Write("What is your choice? ");
                    userInput = Console.ReadKey().KeyChar;
                    Console.WriteLine("\n");

                    validInput = true;

                    // The user chooses to quit the game.
                    if (char.ToUpper(userInput) == 'Q')
                    {
                        QuitGame();
                    }
                    // The user chooses to check their status.
```

(continues on next page)

(continued from previous page)

```
                    else if (char.ToUpper(userInput) == 'D')
                    {
                        CheckStatus();
                    }
                    // The user chooses to hide for the night.
                    else if (char.ToUpper(userInput) == 'C')
                    {
                        StopToFillGas();
                    }
                    // The user chooses to move ahead full speed.
                    else if (char.ToUpper(userInput) == 'B')
                    {
                        MoveAhead(false);
                    }
                    // The user chooses to move ahead slowly.
                    else if (char.ToUpper(userInput) == 'A')
                    {
                        MoveAhead(true);
                    }
                    // The user input was invalid.
                    else
                    {
                        Console.WriteLine("You input was invalid.");
                        validInput = false;
                    }

                    CheckIfCaught();

                }

                if (win)
                {
                    Console.WriteLine("\nCongratulations! You've escaped the police and won the game!");
                }
                else if (!win && !quit)
                {
                    Console.WriteLine("\nYou have lost the game.");
                }
                else
                {
                    Console.WriteLine("\nThanks for playing.");
                }

                Console.Write("Would you like to play again? (Y/N) ");
                userInput = Console.ReadKey().KeyChar;
                Console.WriteLine("\n");

                if (char.ToUpper(userInput) == 'Y')
                {
                    playAgain = true;
                }
```

(continues on next page)

(continued from previous page)

```
                else if (char.ToUpper(userInput) == 'N')
                {
                    playAgain = false;
                }
                else
                {
                    Console.WriteLine("You have entered an invalid value and the game will now close.\n");
                    playAgain = false;
                }
            }

            Console.WriteLine("Thank you for playing. Press any key to exit.");
            _ = Console.ReadKey();
        }

        private static void CheckIfCaught()
        {
            if (char.ToUpper(userInput) != 'Q' && validInput)
            {
                if (gasTankLeft > 8 && !done)
                {
                    Console.WriteLine("Your car ran out of gas and you got caught.");
                    done = true;
                }
                else if (gasTankLeft > 5)
                {
                    Console.WriteLine("Your gas is getting low.");
                }

                if ((milesTraveled - policeMilesTraveled) <= 0 && !done)
                {
                    Console.WriteLine("The police caught you.");
                    done = true;
                }
                else if ((milesTraveled - policeMilesTraveled) <= 15)
                {
                    Console.WriteLine("The police are getting close!");
                }

                if (milesTraveled >= MILES_TO_HIDEOUT && !done)
                {
                    done = true;
                    win = true;
                }
            }
        }

        static void QuitGame()
        {
            done = true;
            quit = true;
```

(continues on next page)

(continued from previous page)

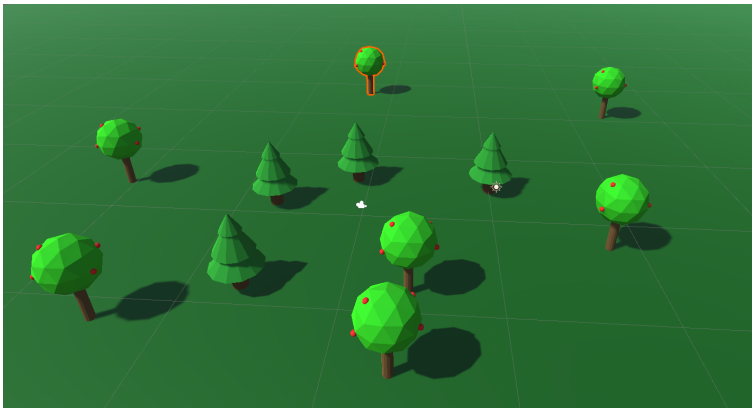
```
        }

        static void CheckStatus()
        {
            Console.WriteLine("Miles traveled: " + milesTraveled);
            Console.WriteLine("Gas fill-ups remaining: " + fillupsLeft);
            Console.WriteLine("The police are " + (milesTraveled - policeMilesTraveled) +
                " miles behind you.");
        }

        static void StopToFillGas()
        {
            gasTankLeft = 0;
            fillupsLeft -= 1;
            policeMilesTraveled += rand.Next(7, 15);

            if (policeMilesTraveled < milesTraveled)
            {
                Console.WriteLine("Your gas tank is full.");
            }
        }

        static void MoveAhead(bool slow)
        {
            int currentMilesTraveled;
            if (!slow)
            {
                currentMilesTraveled = rand.Next(10, 21);
                gasTankLeft += rand.Next(1, 4);
            }
            else
            {
                currentMilesTraveled = rand.Next(5, 13);
                gasTankLeft++;
            }
            milesTraveled += currentMilesTraveled;
            policeMilesTraveled += rand.Next(7, 15);
            Console.WriteLine("You traveled " + currentMilesTraveled + " miles.");

            int findingAHideout = rand.Next(19);

            if (FoundOasis(findingAHideout) && milesTraveled < MILES_TO_HIDEOUT)
            {
                Console.WriteLine("You found an abandoned hideout!");
                fillupsLeft = 3;
                gasTankLeft = 0;
            }
        }

    }
}
```

CHAPTER

TWO

BLENDER TO UNITY

Our goal is to learn to create simple, low-poly 3D items in Blender. Color them. Then import to Unity.



Note: Don't forget to Blender scale interface up before showing this tutorial so people can see.

2.1 Set Up Unity

- Create a new 3D project in Unity
- Create a 20x20 dark green plain for the ground

2.2 Create 3D Items in Blender

- Open Blender
- Notice window and hierarchy, like Unity
- Navigation
  - Click-left to select
  - Middle click to rotate around focused object (Unity is alt-left click)
  - Number pad . to change focus (Unity is F key)
  - Shift-Middle button pans (unity is just left click if you are in the 'hand tool' mode)
  - Show axis thing in upper right to select side views. Also show num pad

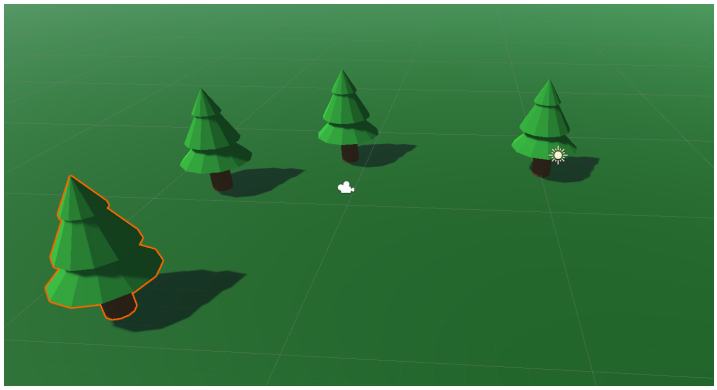
- Delete everything. We don't want to import a camera or light.
- We will be creating one file for each object.
  - Select item in hierarchy or screen, then delete key
- Add->Mesh->Cylinder
  - Lower left, expand out window and select 0.5 meters and 12 verts
- Edit object
  - Explain object mode, and edit mode. Use tab to switch
  - Show vert, edge, face select tools
  - Show alt-click to get circle
  - Show 'E' to extend.
  - Show 'S' to scale.
  - Show 'G' to move.
  - Show xyz to select axis
  - Make pine tree. Show how to scale to zero.

2.3 Materials in Blender

- Materials
  - Show how to create a material for leaves
  - Assign it.
  - Can't see it! Show select material view. And other views.
  - Create new material for trunk
  - Now need to assign. Show wireframe, face select, hidden faces.

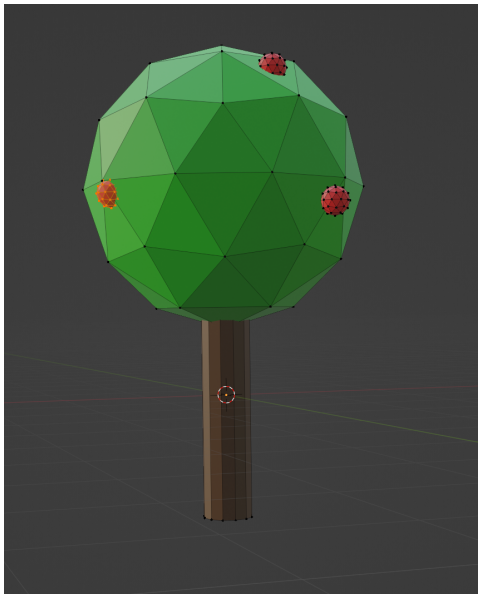
2.4 Import in Unity

- Back to unity
  - Create folder for blender models
  - Open in explorer. Copy path
- Back to blender, save to path
  - pine tree
- Unity & blender native files
  - Export to FBX
- Back to Unity.
  - Drag file into scene



2.5 More Practice

- Repeat, but create a tree using an icosphere. Add apples.
  - Show Ctrl-L for selecting linked
  - Shift-D to duplicate
  - Brand new blender file, do not combine
  - Watch scale



2.6 Weekend Assignment

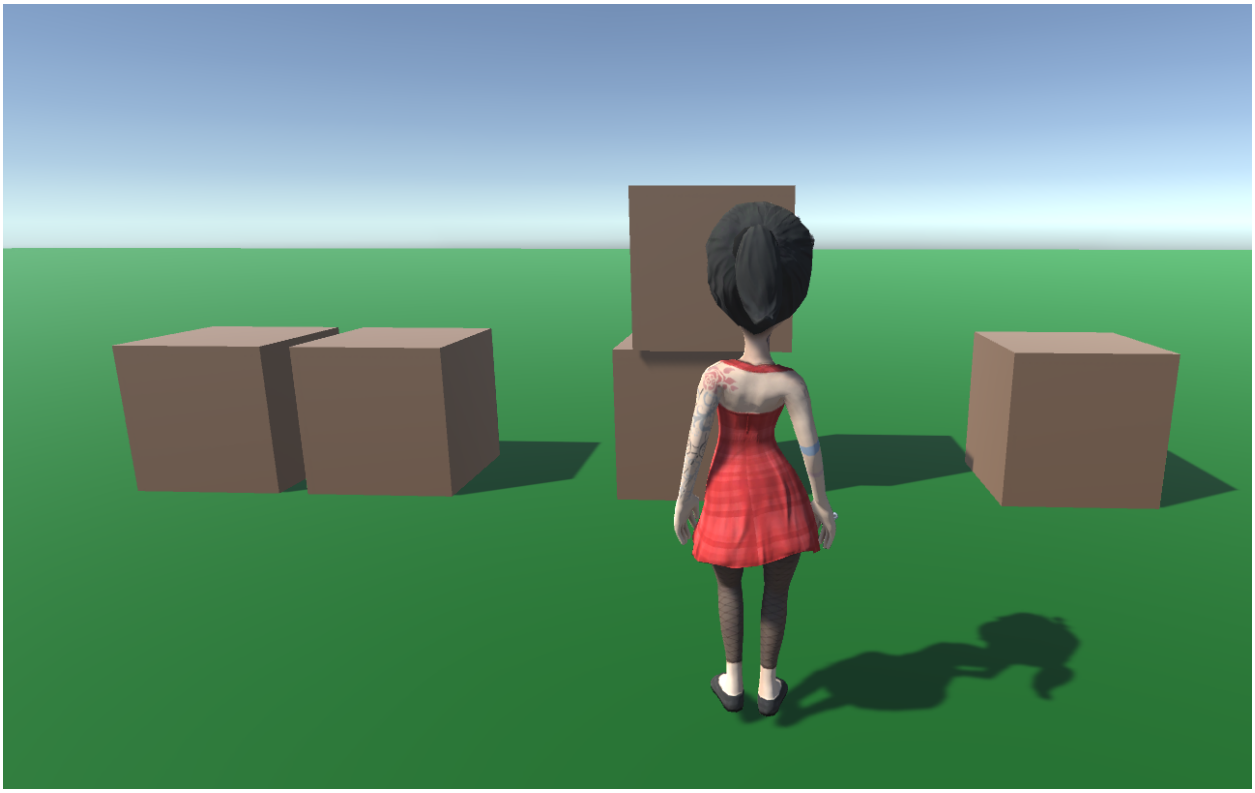
- Out of class, work through Chapter 1 and Chapter 2. You can skip the last object modifiers item in Chapter 2.
  - https://cgcookie.com/course/basics-intro-to-blender-3-0

CHAPTER

THREE

MIXAMO TO UNITY

This covers how to get a 3D character into your scene, using Mixamo character assets and animations.



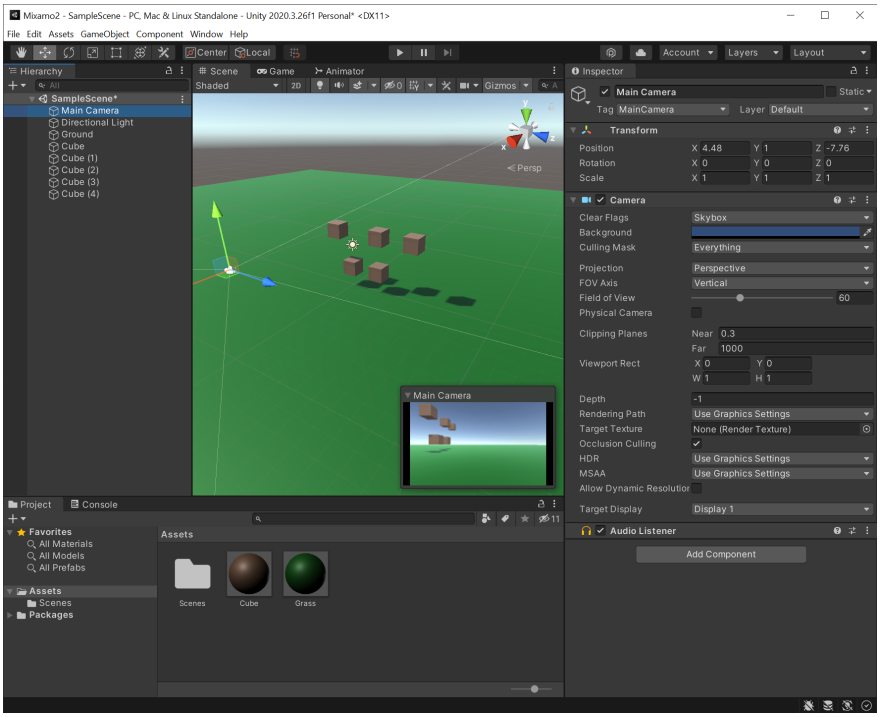
Mixamo has a few character assets (not its primary purpose) and a lot of animations for characters (its primary purpose).

3.1 Setup

Our goal here is to create a landscape for our character to walk around. We'll add a plane and have a few cubes to help with a sense of distance and perspective.

1. Create a new Unity 3D project
2. Add a plane, name it "Ground"
3. Scale x/z to 10x10
4. Create a grass material color

5. Add material to plane
6. Create a cube
7. Create a different material and add to cube
8. Add rigid body physics to cube. Test.
9. Duplicate a few cubes
10. Position the camera
11. Don't forget to save

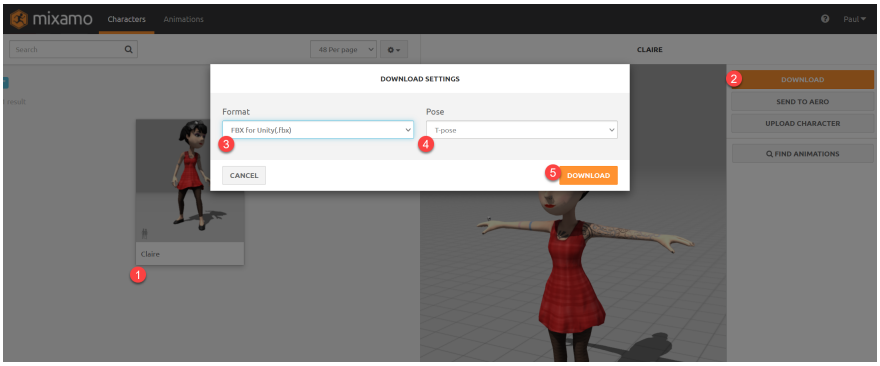


3.2 Download the Character from Mixamo

1. Go to Mixamo.com
2. Log in. You'll log in with an Adobe id or some SSO choice they have.

3.2.1 Get the Character

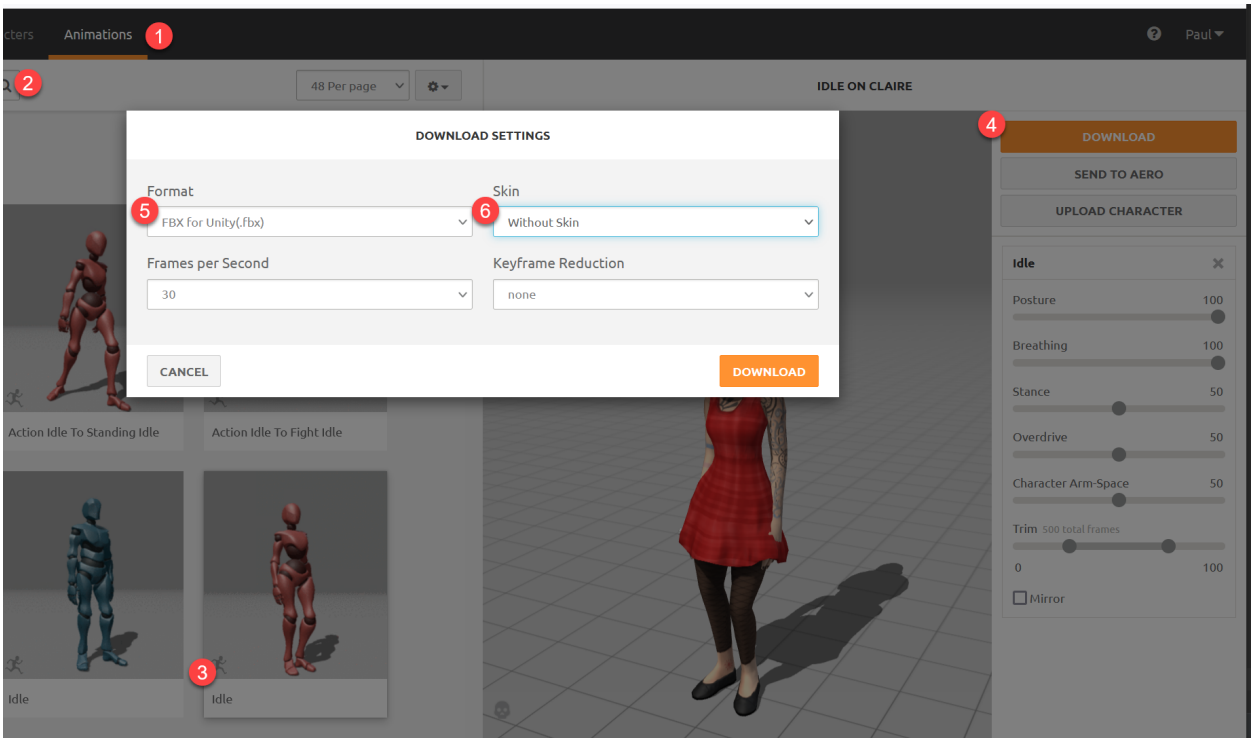
Once there, go to the "Character" tab and find a character you like. I'm using Claire.



1. Select your character
2. Hit “Download”
3. You want “FBX for Unity”. You do **not** want the generic FBX it defaults to.
4. Make sure T-Pose is selected
5. Download

3.2.2 Get the Animations

Now we need our idle and walking animations.



1. Switch to “Animations”
2. Search on “Idle”
3. Select an idle animation. If you don’t see it play with your character hit “refresh” on the browser. You can adjust the animation. For example, widen out the hands so they don’t clip through the characer.

4. Click “Download”
5. Select FBX for Unity. (Again, the default FBX doesn’t work.)
6. Select “Without Skin” because we already downloaded that.

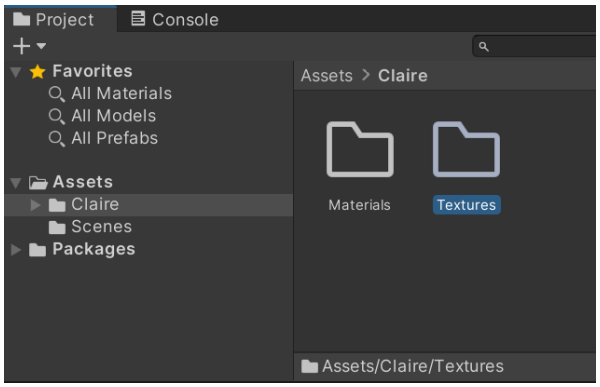
Next, repeat for a walking animation. You’ll get an extra check-box for **in-place** which you must check. This will keep the animation from moving the character forward, while the code thinks the character is in the same location.

> **Warning:** You must select “In-Place” checkbox for any moving animation

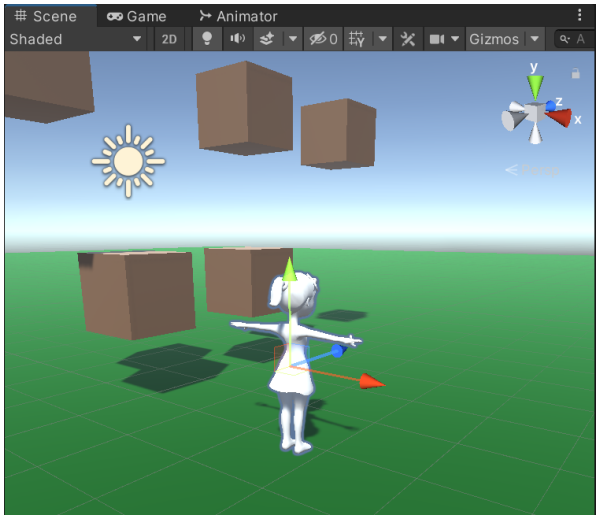
3.3 Add Mixamo Characters and Animations to Project

Now we want to get the character to appear in our project.

1. Create a folder for your character. In this case, I used “Claire”.
2. Create subfolders for “Materials” and “Textures”

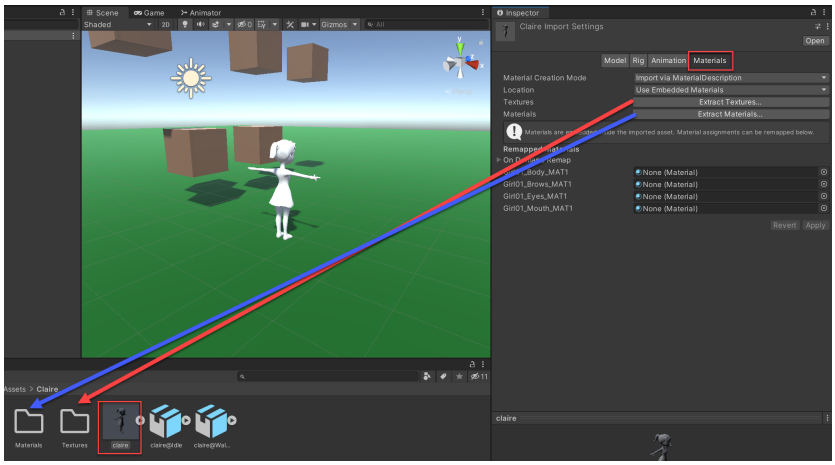


3. Drag the character and two animations from your ‘downloads’ to the folder you created.
4. Drag the character from the assets to your scene. It will be white, as no textures ore materials have been applied yet.

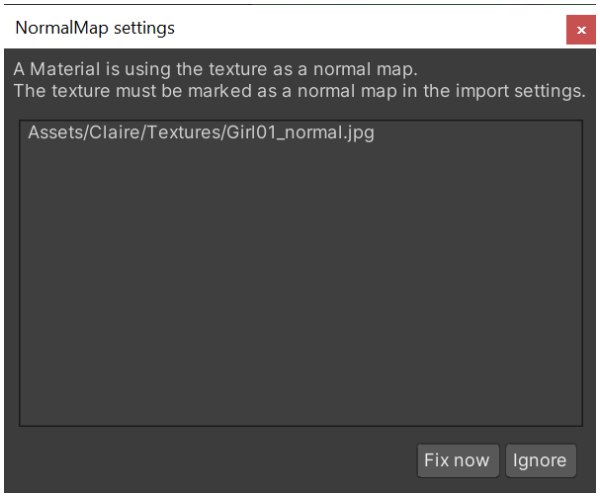


5. Next click on your character in Assets.

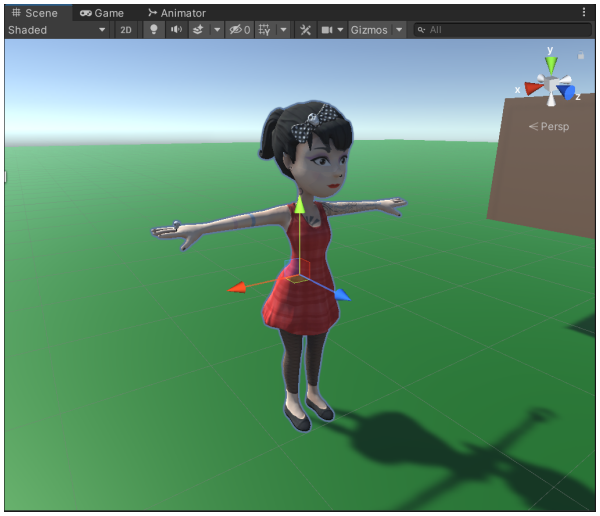
6. Select Materials in the Inspector panel.
7. Click “Extract Textures” and put them in the Textures folder we created.
8. Click “Extract Materials” and put them in the Materials folder we created.



9. If you get a message like this, just go ahead and fix.



10. Now your character should look good.

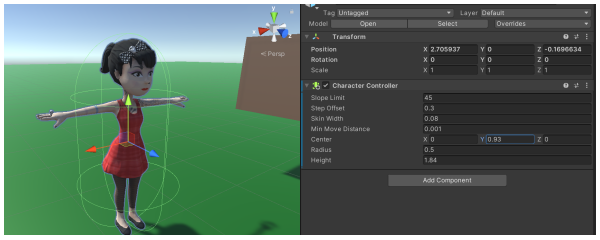


3.4 Get Character to Move

Now we need to get the character to move around. We are going to use a **character controller**. It is more complex than rigid body physics, but offers more control.

3.4.1 Add Character Controller

1. Select your character.
2. Select "Add Component" in the inspector.
3. Add a character controller.
4. The character controller has a 'capsule' for hitbox calculations. Adjust the size and positioning of this so it goes around your character.



3.4.2 Add Character Script

- Make the camera a 'child' of the player and position behind the player.
- Add this character script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CharacterScript : MonoBehaviour
```

(continues on next page)

(continued from previous page)

```
{
    [SerializeField] Transform playerCamera = null;
    [SerializeField] float mouseSensitivity = 3.5f;
    [SerializeField] float walkSpeed = 6.0f;
    [SerializeField] float gravity = -13.0f;
    [SerializeField] [Range(0.0f, 0.5f)] float moveSmoothTime = 0.3f;
    [SerializeField] [Range(0.0f, 0.5f)] float mouseSmoothTime = 0.03f;

    [SerializeField] bool lockCursor = true;

    float cameraPitch = 0.0f;
    float velocityY = 0.0f;
    CharacterController controller = null;

    Vector2 currentDir = Vector2.zero;
    Vector2 currentDirVelocity = Vector2.zero;

    Vector2 currentMouseDelta = Vector2.zero;
    Vector2 currentMouseDeltaVelocity = Vector2.zero;

    void Start()
    {
        controller = GetComponent<CharacterController>();
        if (lockCursor)
        {
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
        }
    }

    void Update()
    {
        UpdateMouseLook();
        UpdateMovement();
    }

    void UpdateMouseLook()
    {
        Vector2 targetMouseDelta = new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis(
↪"Mouse Y"));

        currentMouseDelta = Vector2.SmoothDamp(currentMouseDelta, targetMouseDelta, ref␣
↪currentMouseDeltaVelocity, mouseSmoothTime);

        cameraPitch -= currentMouseDelta.y * mouseSensitivity;
        cameraPitch = Mathf.Clamp(cameraPitch, -90.0f, 90.0f);

        playerCamera.localEulerAngles = Vector3.right * cameraPitch;
        transform.Rotate(Vector3.up * currentMouseDelta.x * mouseSensitivity);
    }

    void UpdateMovement()
```

(continues on next page)

(continued from previous page)

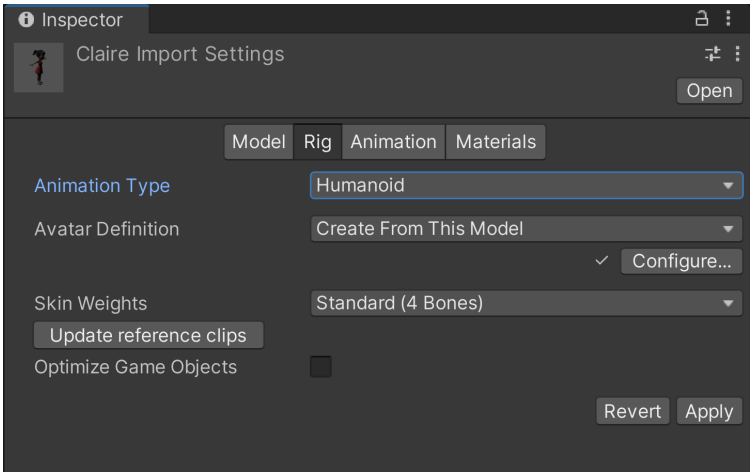
```
    {
        Vector2 targetDir = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));
        targetDir.Normalize();

        currentDir = Vector2.SmoothDamp(currentDir, targetDir, ref currentDirVelocity, moveSmoothTime);

        if (controller.isGrounded)
            velocityY = 0.0f;

        velocityY += gravity * Time.deltaTime;

        Vector3 velocity = (transform.forward * currentDir.y + transform.right * currentDir.x) * walkSpeed + Vector3.up * velocityY;

        controller.Move(velocity * Time.deltaTime);

    }
}
```

- While the character does not animate yet, it should be able to move with mouse and WASD keys.

3.5 Animate

3.5.1 Add Armature Rigs

- Select your character in the assets folder.
- In the “Inspector” tab, select “Rig”.
- Select “Humanoid”
- Select “Create From This Model”.
- Select “Apply”

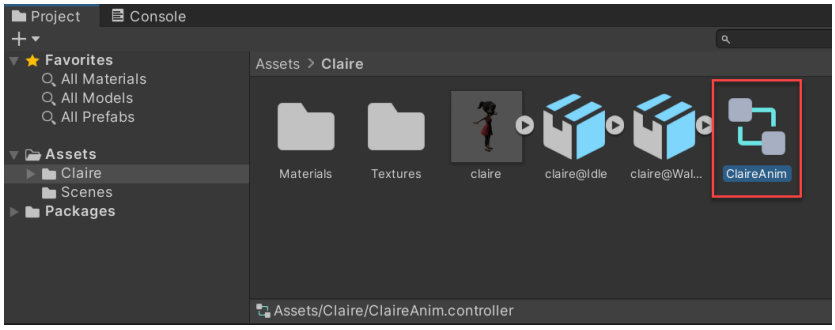


- Select the “Idle” animation.

- In the "Inspector" tab, select "Rig".
- Select "Humanoid"
- Select "**Copy** From Other Avatar".
- Double-click on "Source" and select the avatar you just created
- Select "Apply"
- Repeat for the "Walk" animation.
- There may be warnings. That's ok.

3.5.2 Add Idle Animation

1. Click on your character folder in assets, and add an **Animator Controller**.



2. Double click on the animator controller to edit it. Then drag the *idle* animation to the controller.
3. Drag the animator controller to your player object. Run. The player should now display the idle animation.

3.5.3 Add Speed Parameter

We will need to transition from idle to walking based on speed. We need to update our character controller to spit this out. Here's our updates:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CharacterScript : MonoBehaviour
{
    [SerializeField] Transform playerCamera = null;
    [SerializeField] float mouseSensitivity = 3.5f;
    [SerializeField] float walkSpeed = 6.0f;
    [SerializeField] float gravity = -13.0f;
    [SerializeField] [Range(0.0f, 0.5f)] float moveSmoothTime = 0.3f;
    [SerializeField] [Range(0.0f, 0.5f)] float mouseSmoothTime = 0.03f;
    Animator _animator;

    [SerializeField] bool lockCursor = true;

    float cameraPitch = 0.0f;
    float velocityY = 0.0f;
```

(continues on next page)

(continued from previous page)

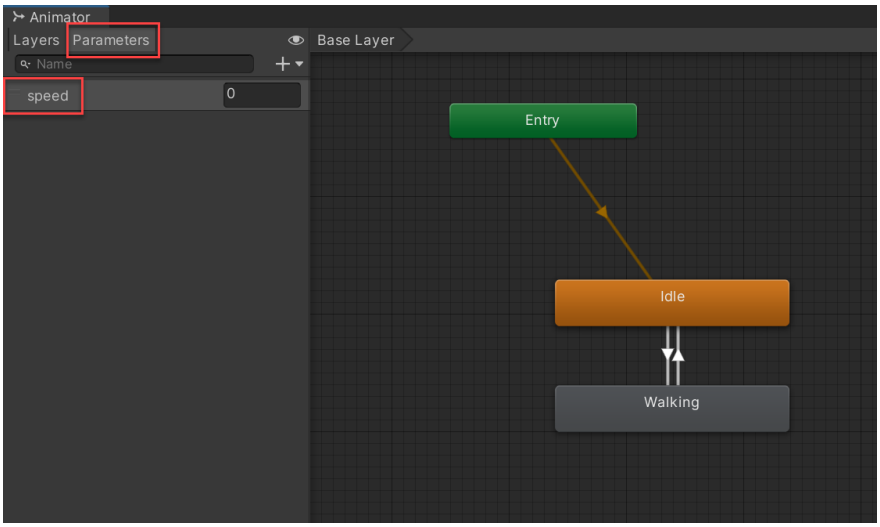
```
    Vector3 velocity = Vector3.zero;

    CharacterController controller = null;

    Vector2 currentDir = Vector2.zero;
    Vector2 currentDirVelocity = Vector2.zero;

    Vector2 currentMouseDelta = Vector2.zero;
    Vector2 currentMouseDeltaVelocity = Vector2.zero;

    void Start()
    {
        _animator = GetComponentInChildren<Animator>();
        controller = GetComponent<CharacterController>();
        if (lockCursor)
        {
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
        }
    }

    void Update()
    {
        UpdateMouseLook();
        UpdateMovement();
        float speedPercent = velocity.magnitude / walkSpeed;
        _animator.SetFloat("speed", speedPercent);
    }

    void UpdateMouseLook()
    {
        Vector2 targetMouseDelta = new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis(
↪"Mouse Y"));

        currentMouseDelta = Vector2.SmoothDamp(currentMouseDelta, targetMouseDelta, ref␣
↪currentMouseDeltaVelocity, mouseSmoothTime);

        cameraPitch -= currentMouseDelta.y * mouseSensitivity;
        cameraPitch = Mathf.Clamp(cameraPitch, -90.0f, 90.0f);

        playerCamera.localEulerAngles = Vector3.right * cameraPitch;
        transform.Rotate(Vector3.up * currentMouseDelta.x * mouseSensitivity);
    }

    void UpdateMovement()
    {
        Vector2 targetDir = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw(
↪"Vertical"));
        targetDir.Normalize();

        currentDir = Vector2.SmoothDamp(currentDir, targetDir, ref currentDirVelocity,␣
↪moveSmoothTime);
```

(continues on next page)

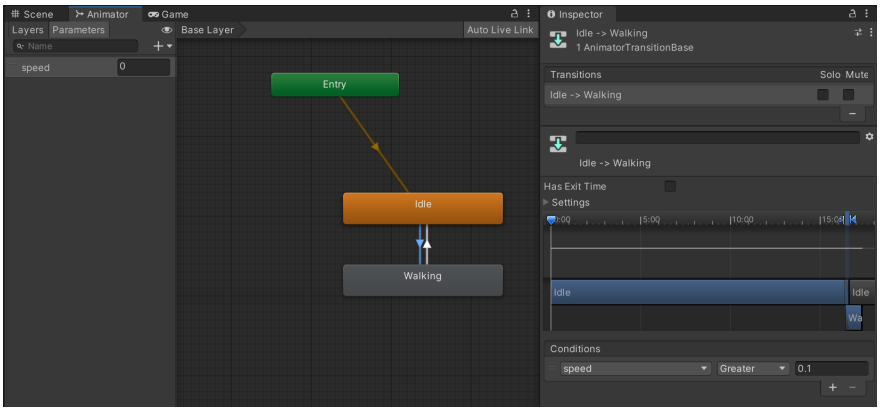
(continued from previous page)

```
67
68          if (controller.isGrounded)
69              velocityY = 0.0f;
70
71          velocityY += gravity * Time.deltaTime;
72
73          velocity = (transform.forward * currentDir.y + transform.right * currentDir.x) *␣
    ↪walkSpeed + Vector3.up * velocityY;
74
75          controller.Move(velocity * Time.deltaTime);
76
77      }
78  }
```

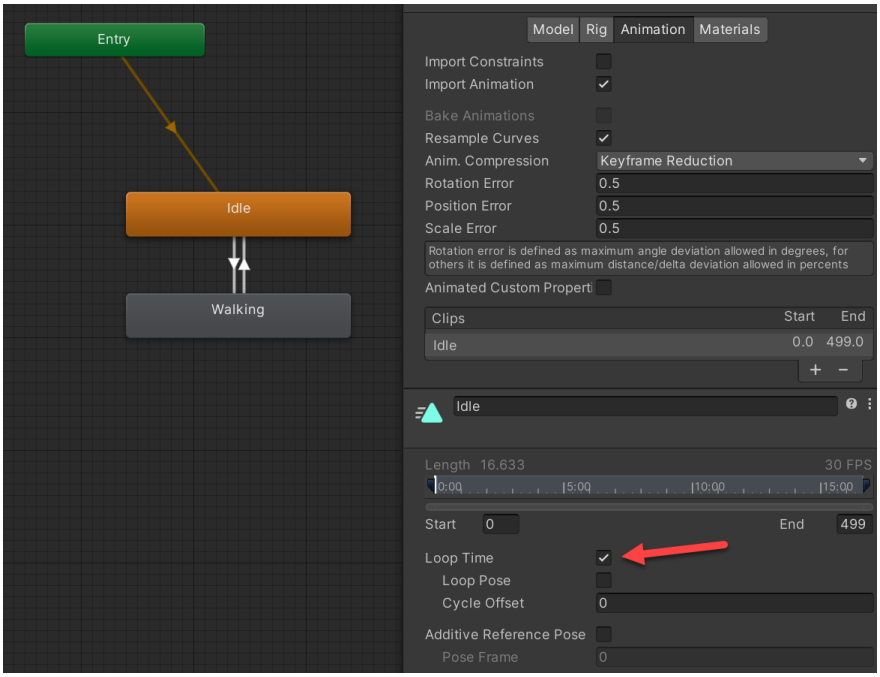
Now in the Animator, we should be able to add `speed`:



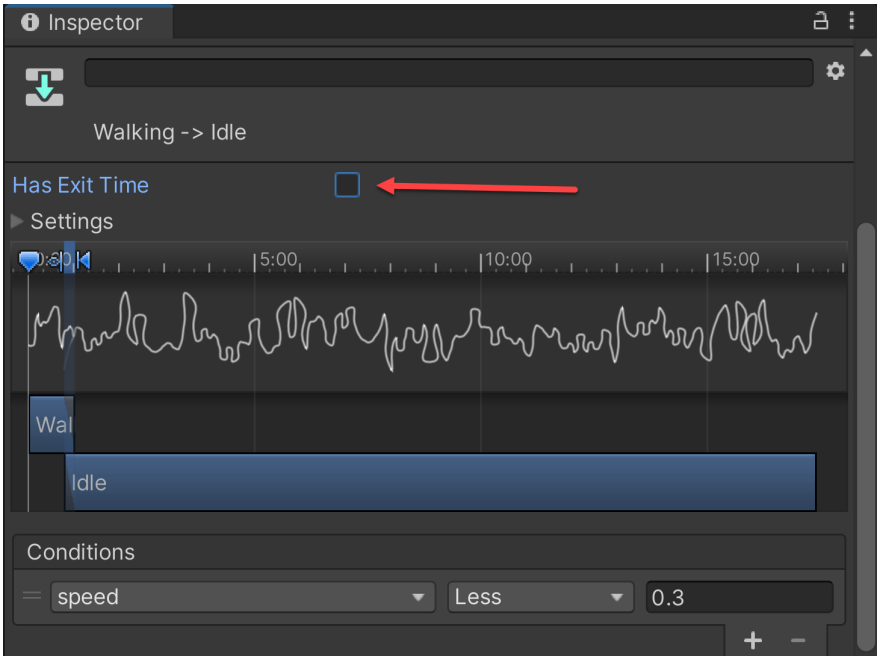
Then we can add in our "walk" animation. Add transitions, and make it based on speed. Greater than 0.3, we animate. Less than 0.3, we idle.



Right now, the animations will only run once. Double-click between both animations and make sure that "Loop Time" box is checked for both animations.

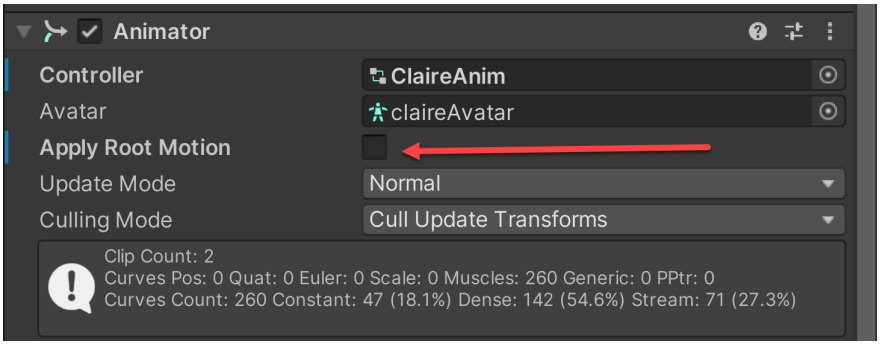


Also, the animations won't transition until they are done. Flip between both animations and *uncheck* "Has Exit Time."



3.5.4 Uncheck Root Motion

Depending on your animation, the animation can move the character. Typically it works best if it is just an animation. Select your character, and in the "Animator" section, uncheck root motion:



3.6 References

- Acacia Developer. First Person Controller. Sep 10, 2020
- Acacia Developer. Unity FPS Controller code. Sep 10, 2020
- Niklas Bergstrand. Adding walk and run animation in Unity. May 19, 2021

CHAPTER

FOUR

TEXTURE OBJECTS

We want to put an image on an object, rather than just have a solid color.

4.1 Texture Types

There are several types of textures.

- Diffuse/Albedo map - Color for object. The is the basics of what you need. Although the image can look "flat." Think bricks. Shouldn't look flat, but will be with just a diffuse map.
- Bump maps - Create illusion of depth via grayscale data. Shade of gray is height. These are grayscale images.
- Normal maps - Better than bump maps, uses RGB for more info. This can give us x, y, and z. Allows for angle and more realistic looks. These maps tend to look blue.
- Displacement/Height map - This map is used to actually changes surface they are on.
- Specular/Metallic - Maps out what part of the image is shiny.

Here are some samples from Texturise, their "Tilable Wood Planks Texture".

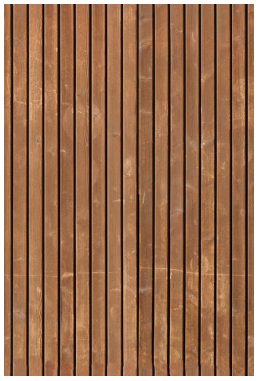


Fig. 1: Texture

Here they are, in action on Blender.

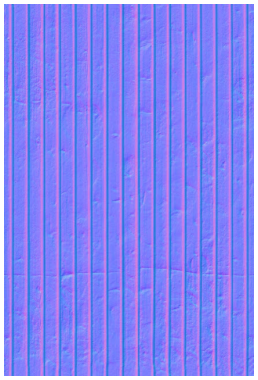


Fig. 2: Normal

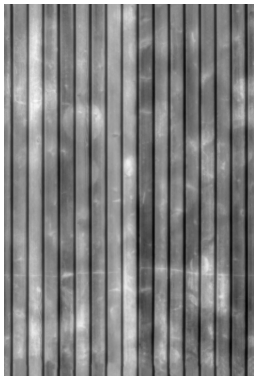


Fig. 3: Specular

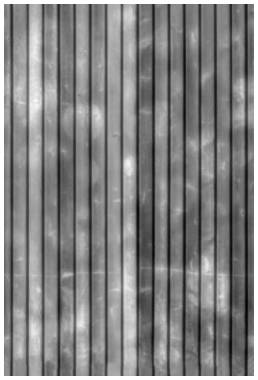


Fig. 4: Displacement

Fig. 5: Albido/Texture image/Color

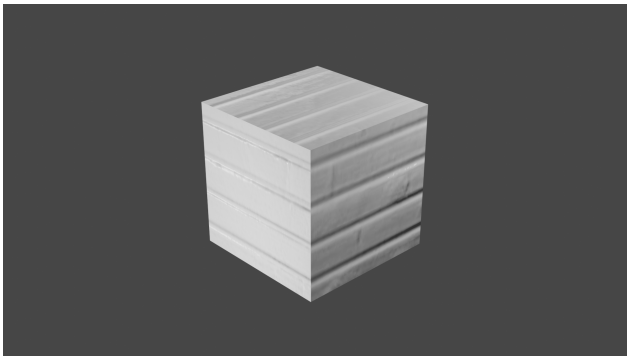


Fig. 6: Normal

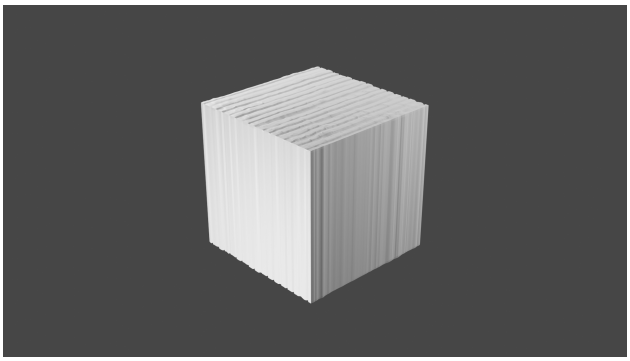


Fig. 7: Displacement

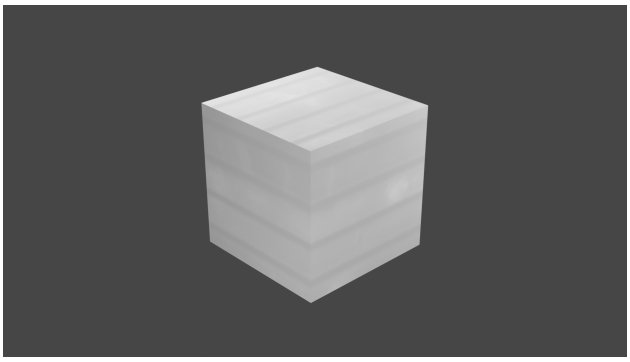


Fig. 8: Specular

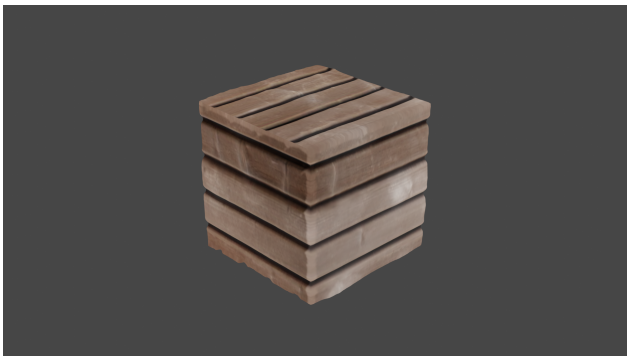


Fig. 9: Everything

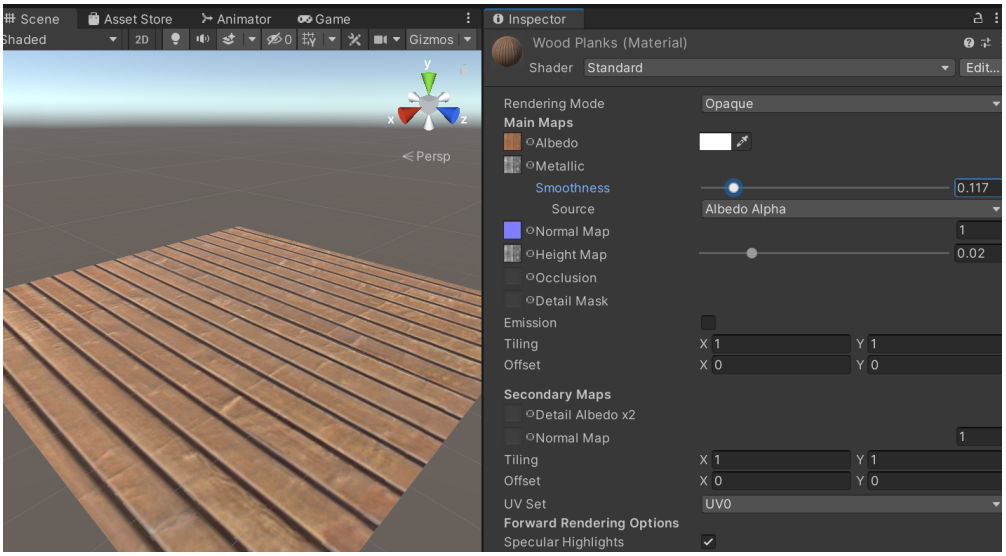
4.2 Texture Websites

Where can you get textures?

- Texturise
- Poly Haven
- Poliigon (Paid)

4.3 Very Simple Textures

- Create a new project.
- Add a 10x10 plane.
- Create a folder called “Textures”
- Toss the images there.
- Create a material in that folder.
- Toss onto the plane.
- Put images into texture
  - Toss ‘texture’ to Albedo.
  - Toss ‘specular’ to ‘metalic’. Change to ‘Albedo Alpha’ and turn smoothness down to about 0.1. You can use this for occlusion instead.
  - Toss ‘normal’ to ‘normal map’
  - Toss ‘displacement’ to ‘Height map’



- You can change the 'tiling' to control how many times it repeats on the surface.

4.4 UV Mapping

Take some road textures:

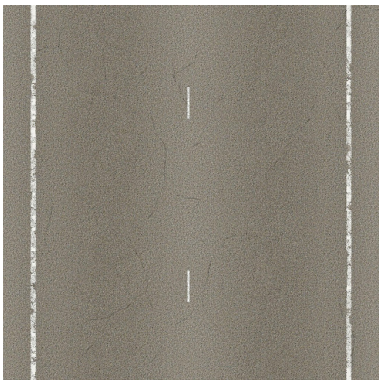


Fig. 10: Road texture

Create a road texture. I used specular for occlusion. Apply to a new cube.

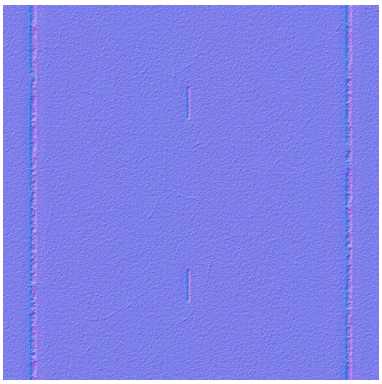


Fig. 11: Road texture normal

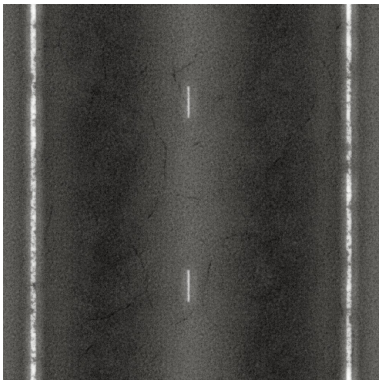
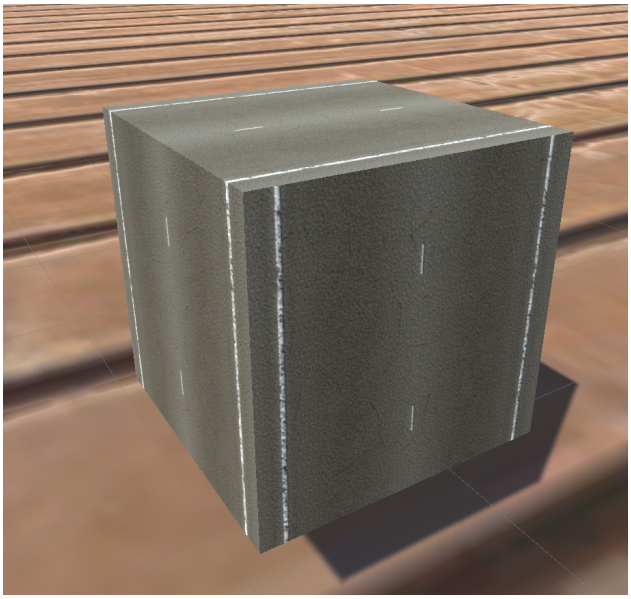
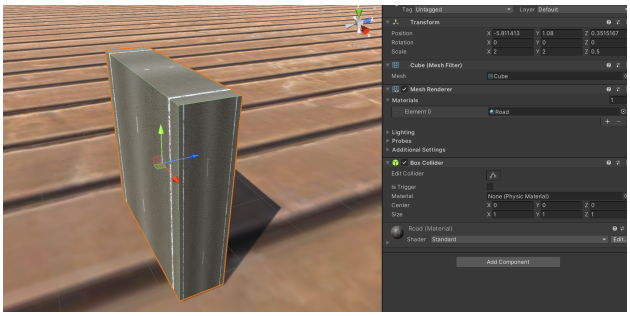


Fig. 12: Road texture specular

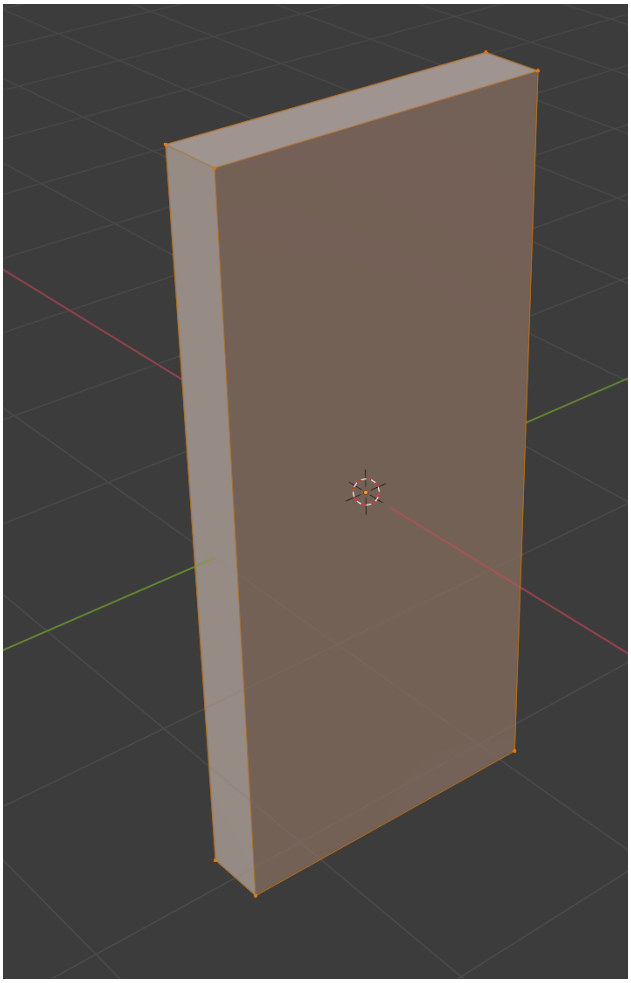


Looks ok. But what if we scale the cube?

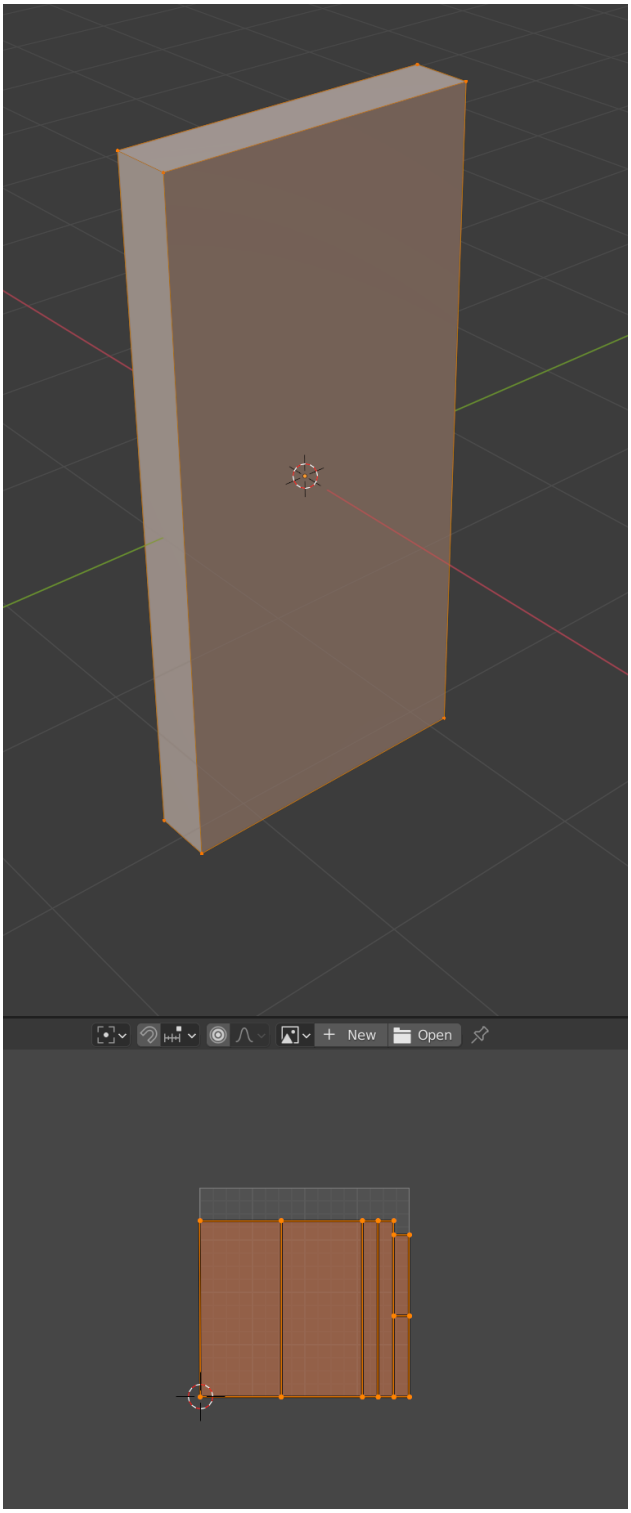


We need to change the geometry, and not scale the item. Then do a “UV Unwrap”.

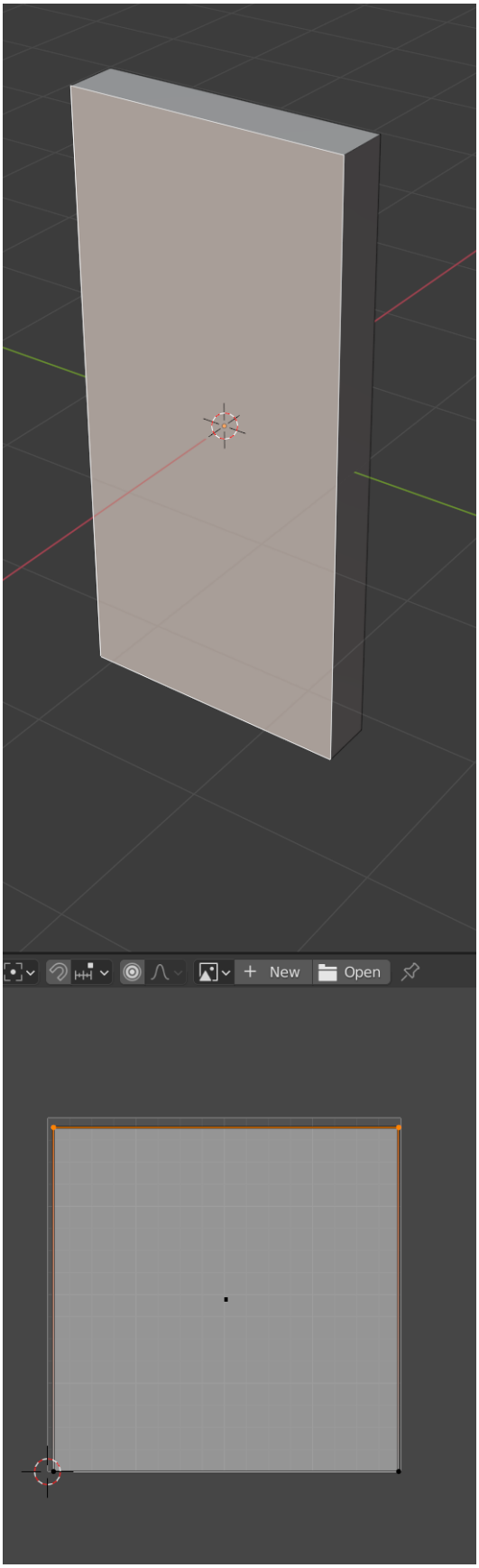
Go to blender. Create a cube. Go into **edit mode** and not object mode. Change the cube dimensions.

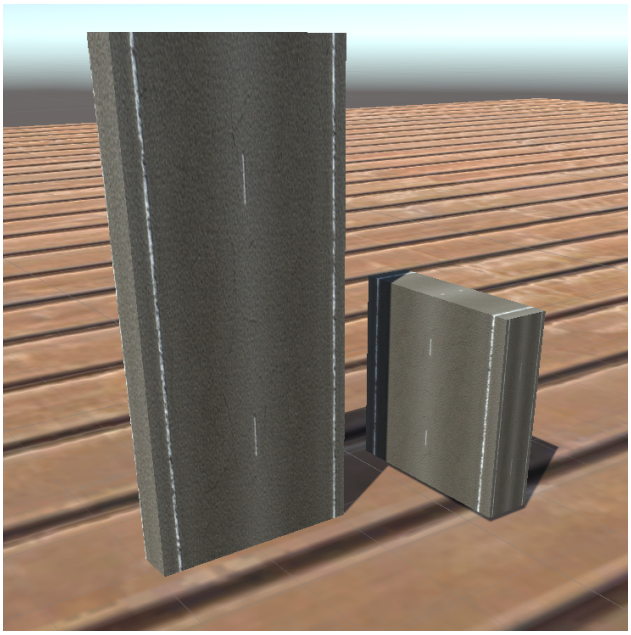


Change the bottom view to UV. Do a smart UV unwrap:



Delete camera and light. Save into your Assets folder. Toss cube onto scene. Apply material. See how it maps? Change mapping. Save. See results.


New
Open



CHAPTER

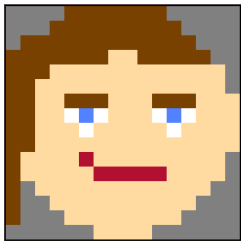
FIVE

2D UNITY PART 1

Contents

5.1 Create sample sprites and add to Unity

1. Clone the base Unity project: https://github.com/pvcraven/2022_Class_2D_Project
2. Create sprites in Aseprite.
   - Use NES palette
   - Create a 16x16 character.



   - Create a 16x32 tree. (Or some other size, keeping in mind 16x16 is the character size.)

- Save to `Assets/Sprites/Trees` or `Assets/Sprites/Characters` folder.
- Call your character `tree_name` or `character_name`. Obviously, use your first and/or last name, not "name".
- Export your sprite as a `.png` in that same folder.

3. Open in Unity, confirm the assets are there.
4. Do a git add, commit, push and pull to sync with the whole class.

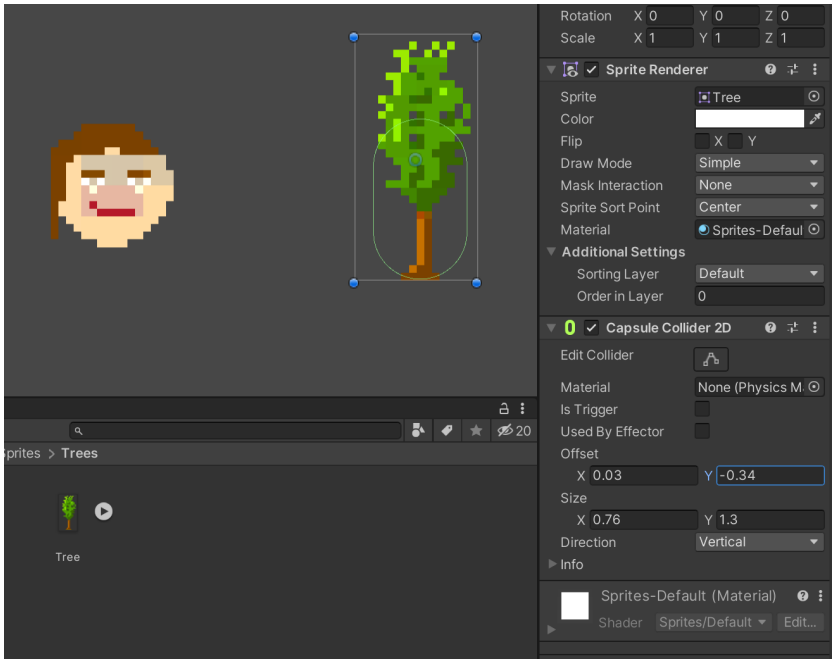
> **Warning:** Be careful of .meta files
>
> Unity adds a `.meta` file that tags a GUID for each file. If you create or move a file into a Unity project, let unity create a `.meta` for it before check in! This includes the exported `.png`. Failure to do this will cause a lot of merge headaches.

5.2 Change sprite settings

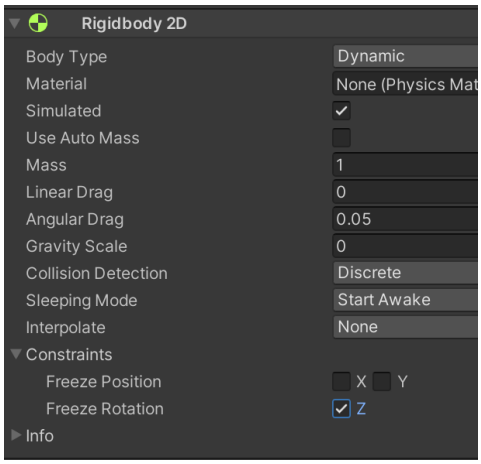
1. Create your own scene. Call it `scene_name`.
2. Drag character onto the screen.
3. Way too small. Unity defaults to 100 pixels to one 'unit' which is 1 meter. Change from 100 to 16.
4. Great. Now the character is blurry. Change the filtering to 'point'.
5. Character might be blotchy. Turn off compression.
6. Should be able to run the scene and see character properly.
7. Repeat these steps for your sprites. Don't do this for other people's sprites.
8. Sync with GitHub.

5.3 Make sprites solid

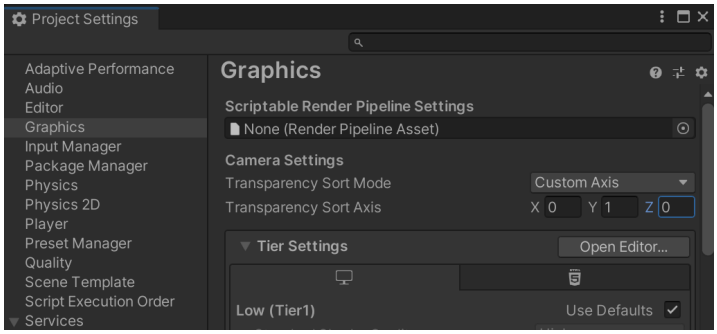
1. Add a rigid body 2d. Run the game. Character should now fall.
2. Zero out the gravity.
3. Add to your character, the MyCharacterController script that is already in the project under the scripts folder. Examine the script and see how it works.
4. Should be able to move character with WSAD. Can adjust speed as needed.
5. Add your tree.
6. Try running. No collision.
7. Add colliders to the character and tree.
    - There are circle colliders, capsule colliders, box colliders. Pick the best one.
    - You might not want to make a collider around everything for a more 3D look.



8. Try running. Character spins!
9. Freeze rotation.



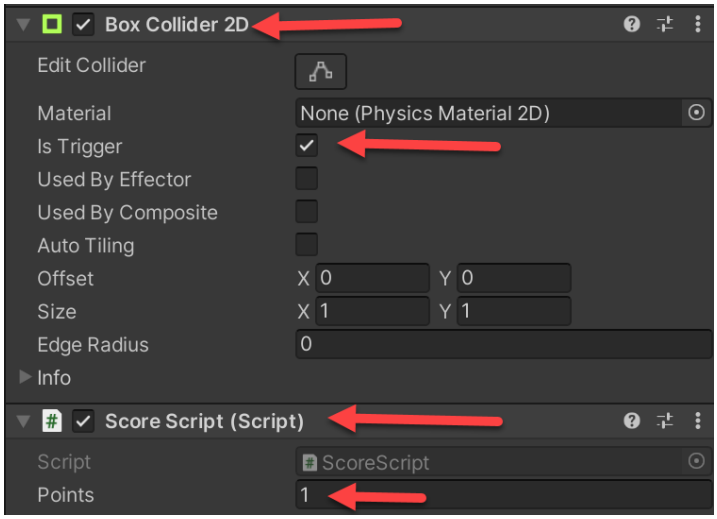
10. Character may or may not appear behind/ahead of the tree properly. You can use sort mode in project settings to fix:



5.4 Add in score

Add in a sprite to increase your score.

- You'll need a collider. Make the collider a "trigger".
- You'll need to add in the ScoreScript. Examine this script and the character controller together to see how they work.
- Set the points for the score script.

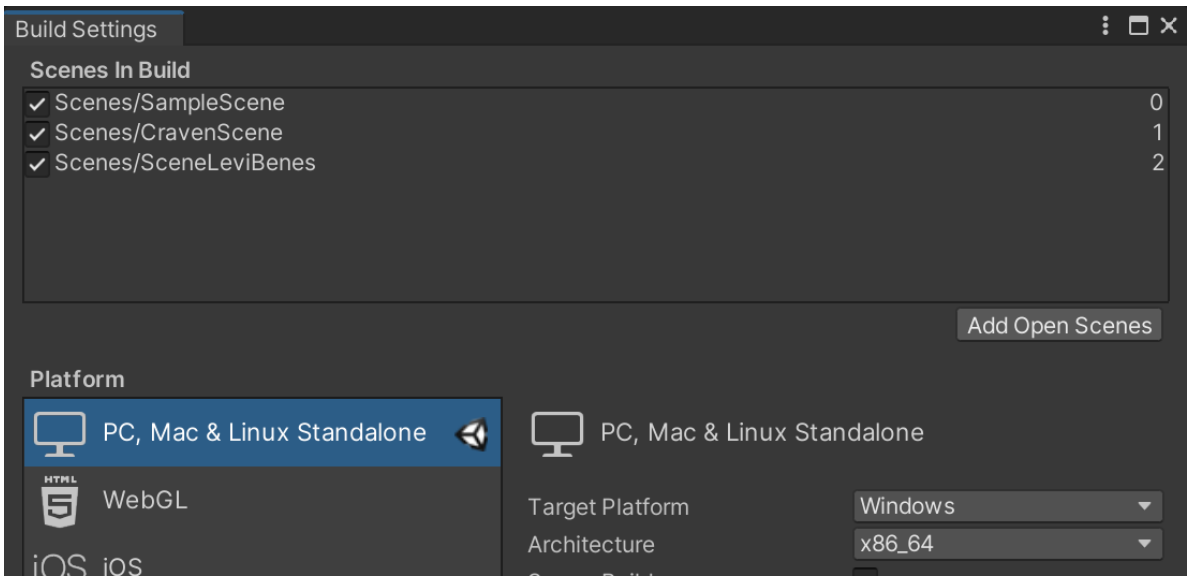


- Test.
- You can also have items that make the score go down by putting in a negative number for points.

5.5 Add in scene change

Create a sprite that will will cause you to go to the next level.

- You'll need a collider. Make the collider a "trigger".
- You'll need to add in the SceneChangeScript. Examine this script and the character controller together to see how they work.
- Your scene must appear in File…Build Settings. This is where you determine the order of levels. As this is a common area, only one person can edit at a time. So let the instructor do this in class.



5.6 Summary

This should step you through most of what you need to complete *2D Assignment 1*. Expand what you've learned to create an explorable level. Don't worry about the background image yet, we'll get to that with tiles.

CHAPTER

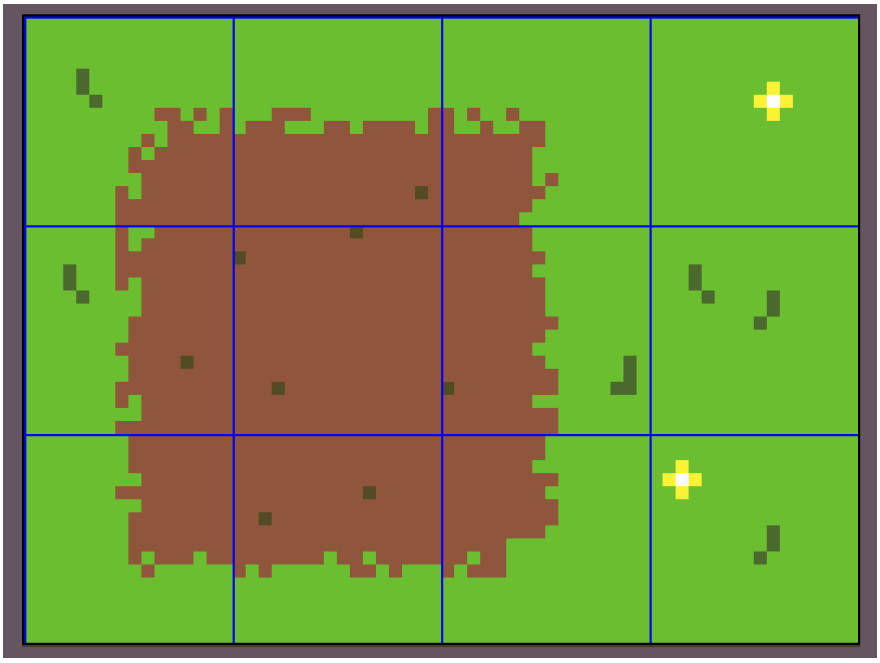
SIX

2D UNITY PART 2

Contents

6.1 Create tile set

Tiles will be 16x16. We'll make multiple tiles at a time. Make a 16*3 and 16*4 image:

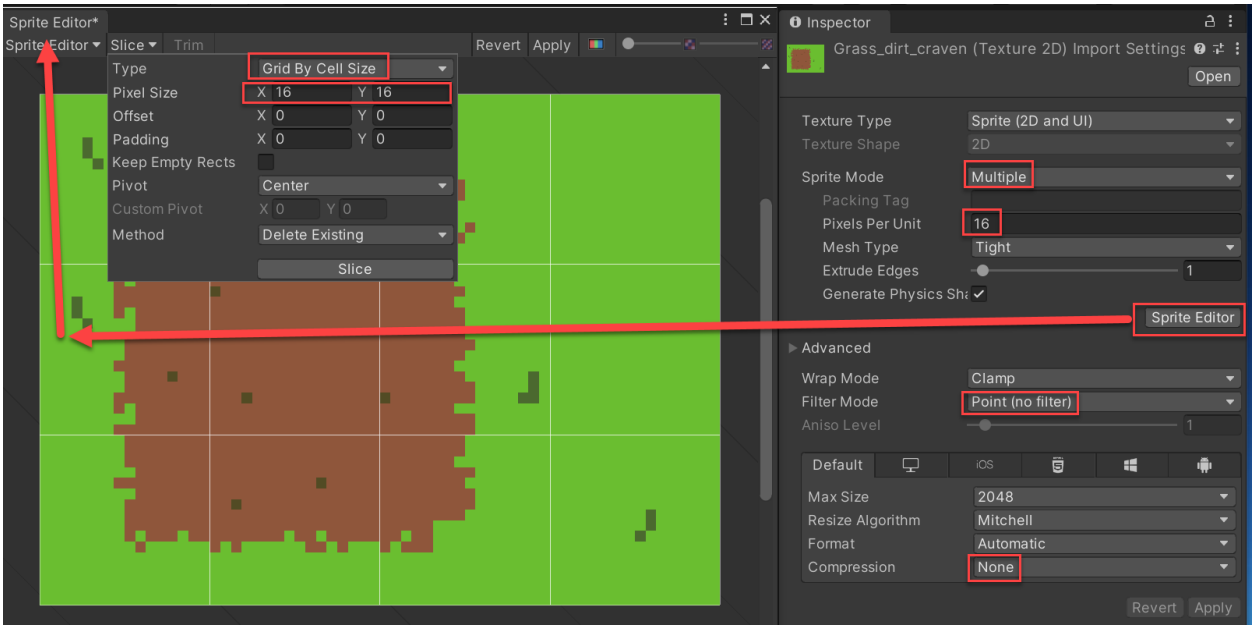


Keep in mind in Aseprite you can:

- Use things like 16*3 in the sprite dimensions, no need to multiply itself.
- You can show the grid overlay

6.2 Import and split tile set

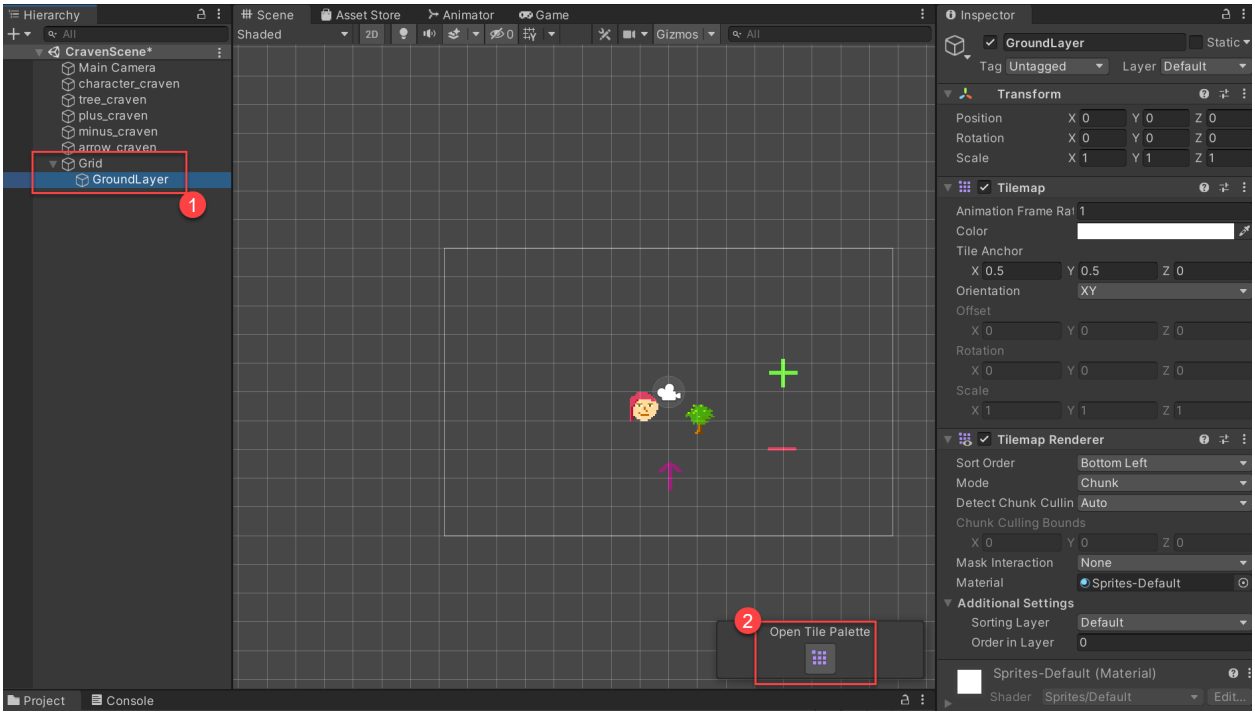
When you import the sprite, we need to set our standard three changes, and then set it to multiple sprites. Then we click on the nearly-hidden sprite editor button and slice it up.



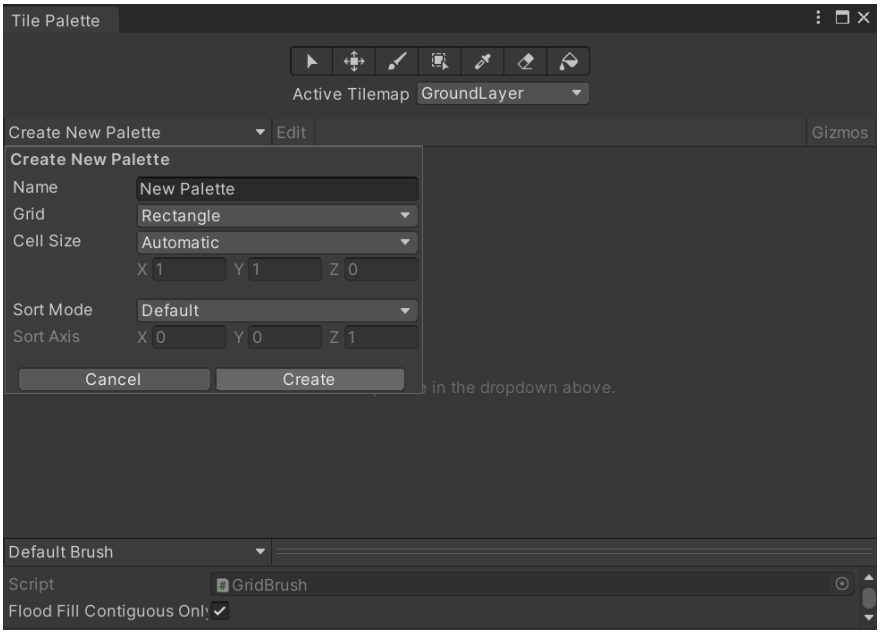
- Commit and push.

6.3 Create tile map and palette

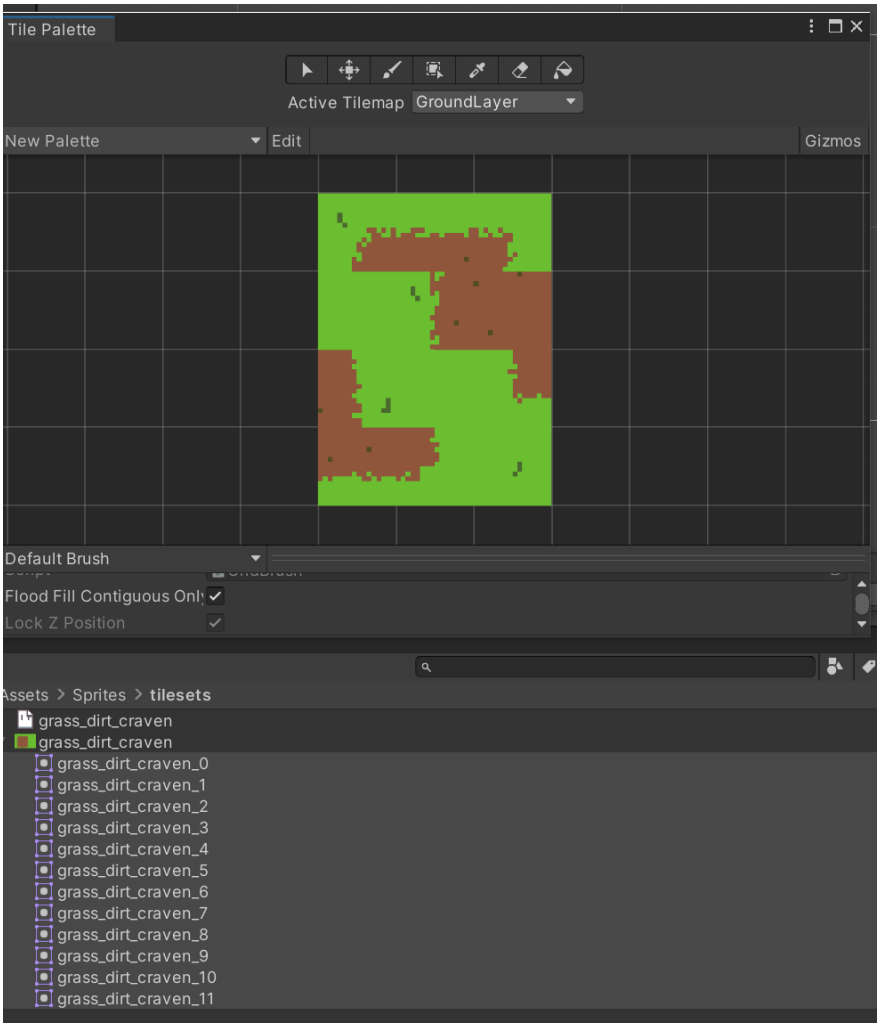
- Create a new rectangular tile map for your scene.
- Open the tile palette.
- Create your own tile palette with your own name



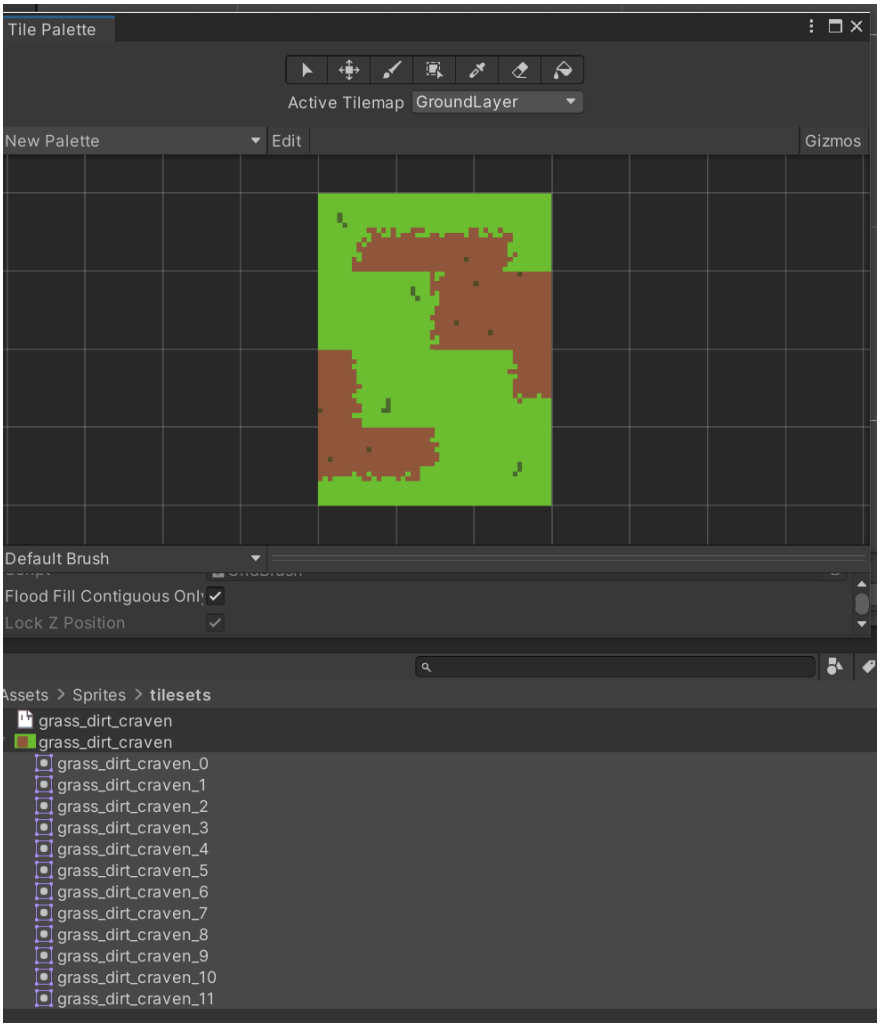
Create a new palette. Create a new folder for it “Tile Palette”.



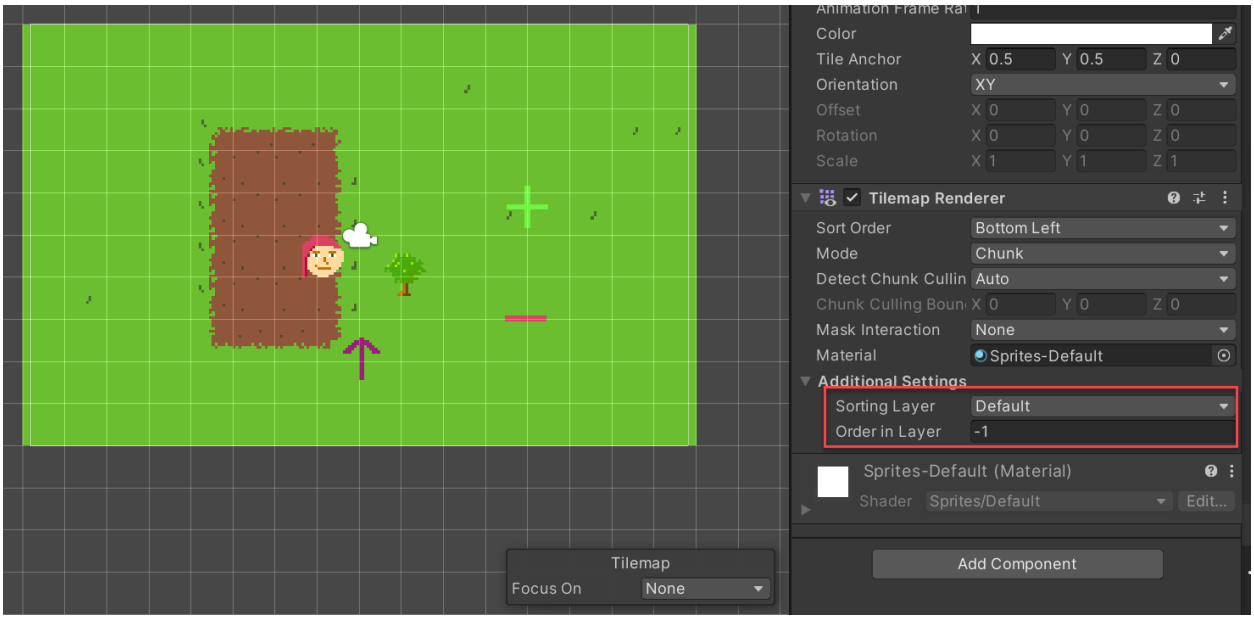
Select your sprites. Move to palette. Create folder for “Tile images”.



Order is weird. Somehow there's a way to import better I think, but I don't know it. To change order, click 'Edit' button and then alternate between S and M keys to move tiles to where you'd like.



- Paint with the tiles.
- Change your rendering order so tiles appear below your sprites. Use layers, or ordering in layers.



- Show how to do layers
- Show how to do a tile collider 2d

CHAPTER

SEVEN

2D ANIMATION

Contents

In this tutorial we'll work on animating sprites.

7.1 Create a time-based animation in Aseprite

Create a folder for your animation.

Follow one of these tutorials:

Fig. 1: Source: SadFace-RL Fire Animations

Fig. 2: Source: SadFace-RL Water Animations

Work on using:

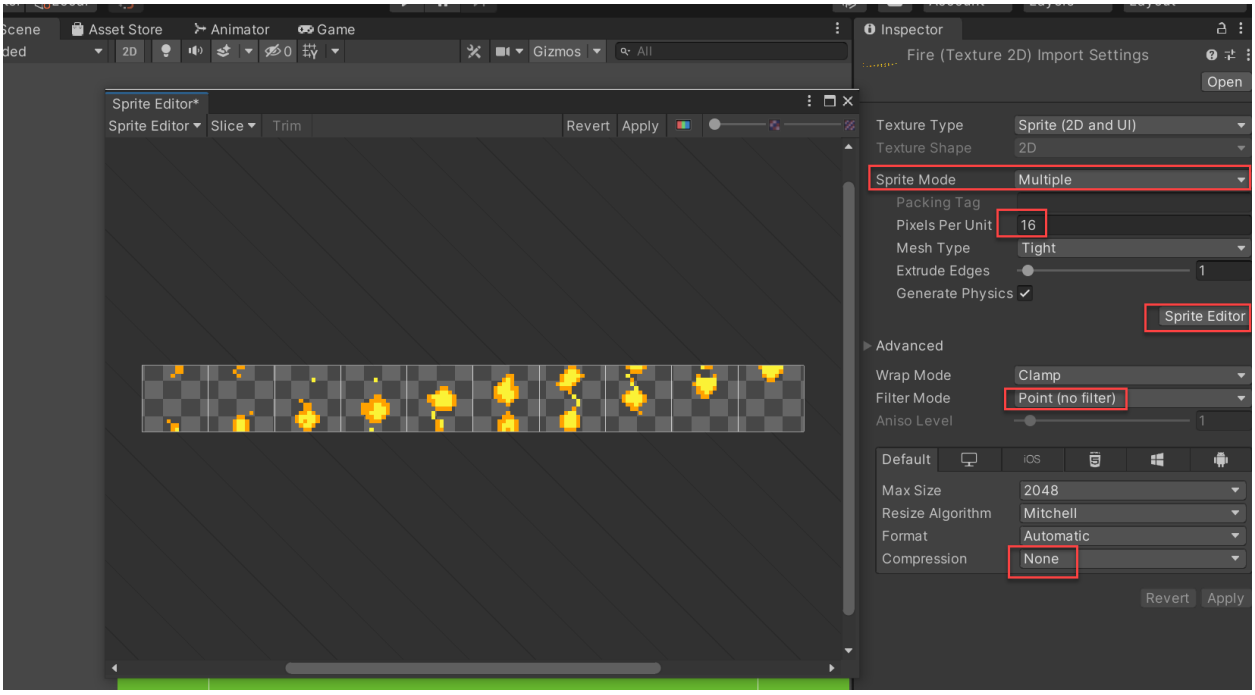
- Keyboard shortcuts
- Select tool
- Frames

Export a sprite sheet.

- File->Export Sprite Sheet
- Output->Output file

7.2 Import a sprite sheet in Unity

Import a sprite sheet and slice it like we did before.



- Drag the first image onto your scene.
- Click Window…Animation
- Click your object, you should see an option to create an animation and controller from it.
- Drag images onto the timeline
- Too fast.
- Drag out the frames, slow it down

Fig. 3: Source: SadFace-RL Animation, getting started

7.3 Create animated character frames in Aseprite

Fig. 4: Source: SadFace-RL Characters, the human male

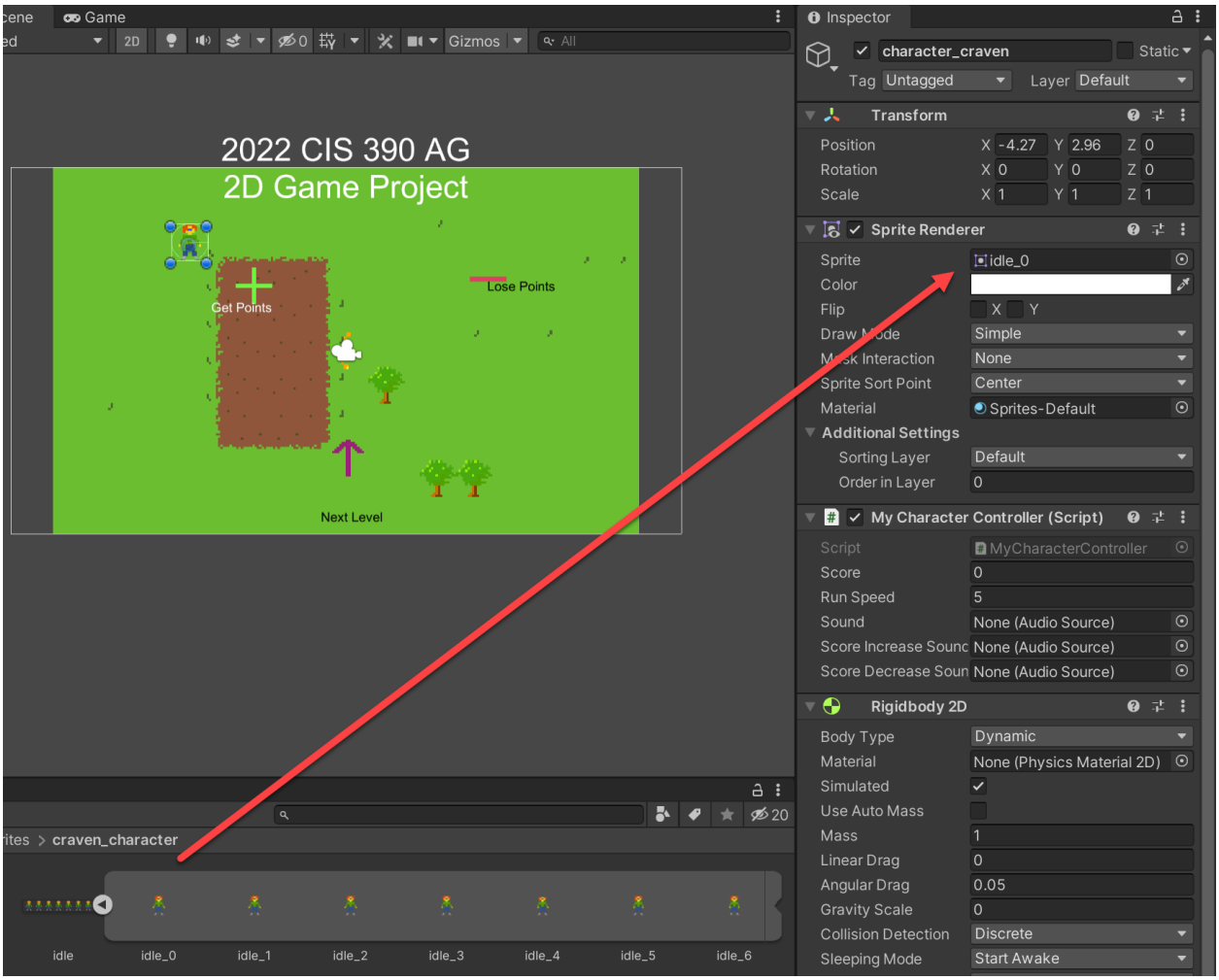
Fig. 5: Source: SadFace-RL Animation, the walk cycle

7.4 Get character working with idle animation in Unity

Here's a video that covers what we are doing:

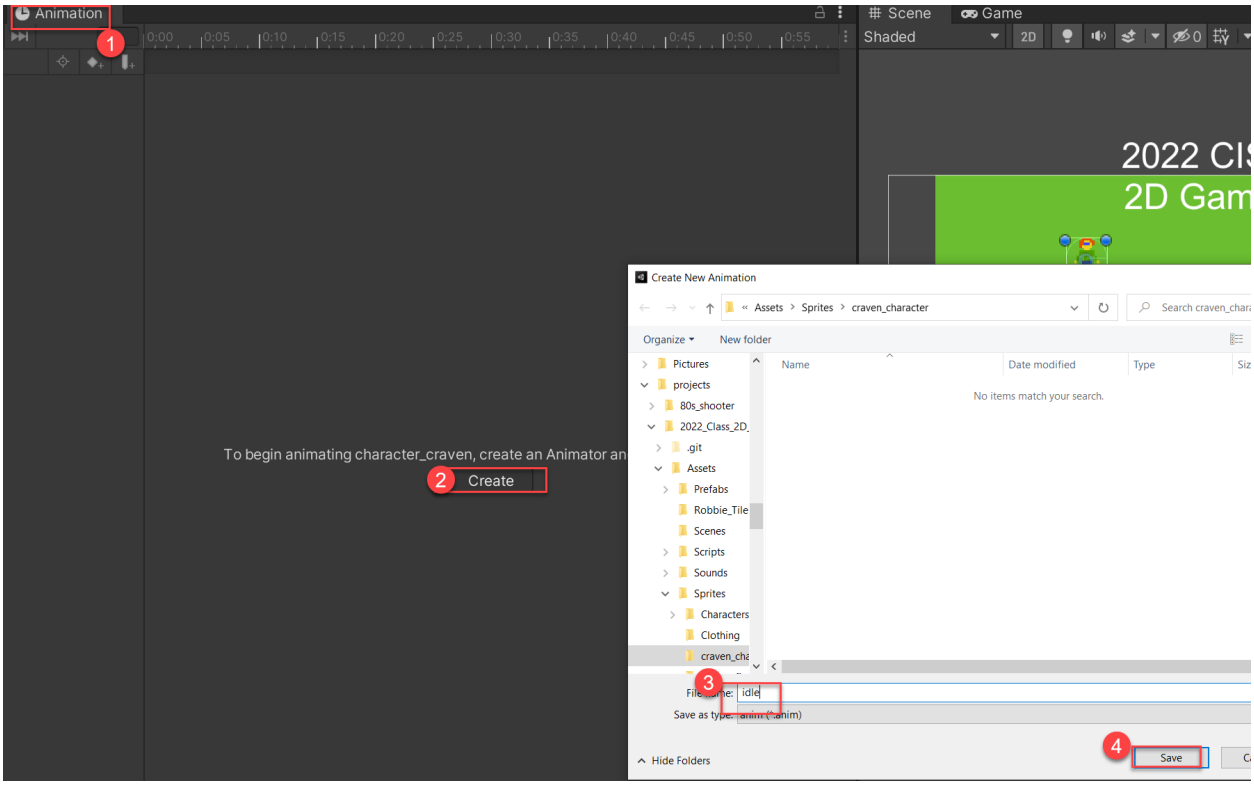
First, go ahead an import your character animations, then slice up the images.

If you want to replace a character you already have with the animated sprites, (recommended) you can replace the character's texture by dragging the sprite image to the proper location.

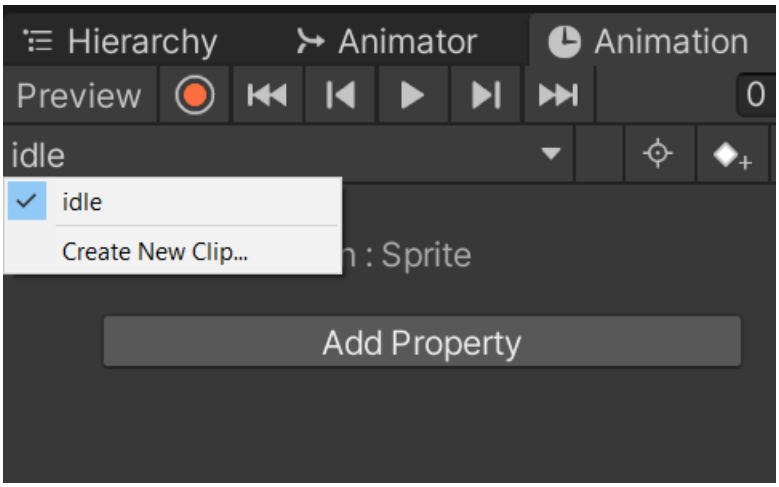


Make sure your program still works ok.

Create an idle animation for your character like we did before.



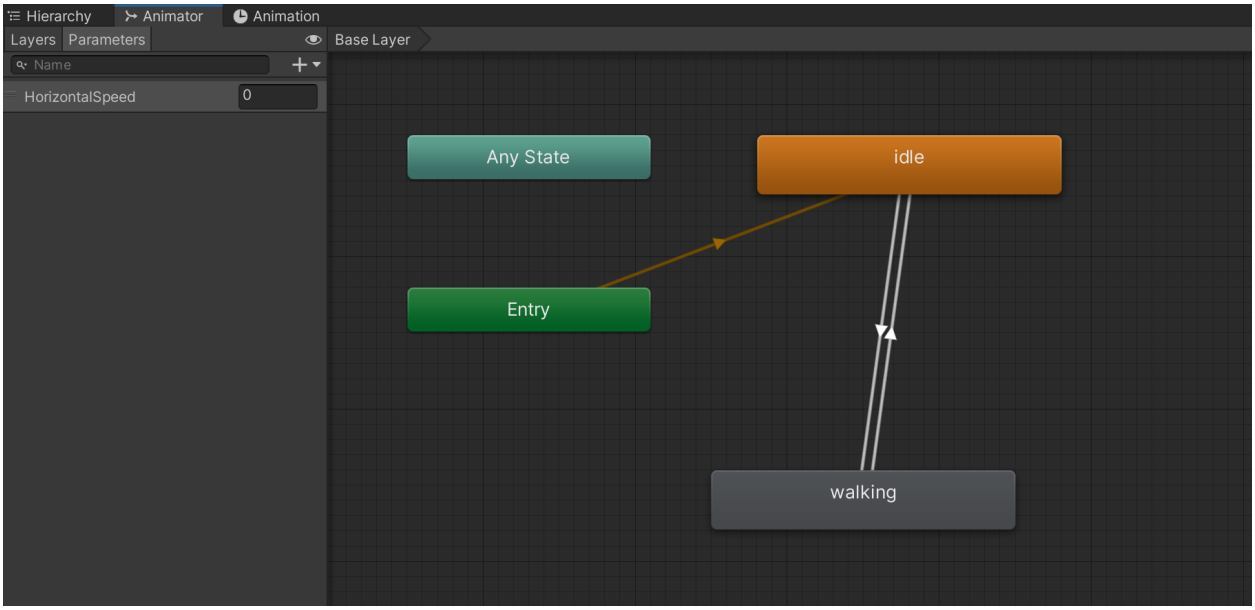
Make sure that works. Now we need a clip for walking/running. Add a new clip from the Animator tab:



Show how to play clip, and change clip.

See how both clips show up in Animator.

Add a parameter, and transitions:



Update code:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class MyAnimatedCharacterController : MonoBehaviour
{
    public int score = 0;

    Rigidbody2D body;


    float horizontal;
    float vertical;
    float moveLimiter = 0.7f;

    public float runSpeed = 5.0f;

    public AudioSource sound;
    public AudioSource scoreIncreaseSound;
    public AudioSource scoreDecreaseSound;

    private SpriteRenderer spriteRenderer;
    private Animator animator;

    void Start()
    {
        // Get the rigid body component for the player character.
        // (required to have one)
        body = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
```

(continues on next page)

(continued from previous page)

```
        animator = GetComponent<Animator>();
    }

    void Update()
    {
        // Get our axis values
        horizontal = Input.GetAxisRaw("Horizontal");
        vertical = Input.GetAxisRaw("Vertical");
    }

    void FixedUpdate()
    {

        // If player is running diagonally, we don't want them to move extra-fast.
        if (horizontal != 0 && vertical != 0) // Check for diagonal movement
        {
            // limit movement speed diagonally, so you move at 70% speed
            horizontal *= moveLimiter;
            vertical *= moveLimiter;
        }

        if (horizontal > 0.1)
            spriteRenderer.flipX = false;
        else
            spriteRenderer.flipX = true;

        // Set player velocity
        body.velocity = new Vector2(horizontal * runSpeed, vertical * runSpeed);
        animator.SetFloat("HorizontalSpeed", Mathf.Abs(horizontal));
    }

    void OnTriggerEnter2D(Collider2D colliderEvent)
    {
        // Did we run into an object that will affect our score?
        ScoreScript scoreObject = colliderEvent.gameObject.
↪GetComponent(typeof(ScoreScript))
                                  as ScoreScript;

        if (scoreObject != null)
        {
            // Yes, change the score
            score += scoreObject.points;


            // Destroy the object
            Destroy(colliderEvent.gameObject);
        }

        // Did we run into an object that will cause a scene change?
        SceneChangeScript sceneChangeObject = colliderEvent.gameObject.
↪GetComponent(typeof(SceneChangeScript))
                                            as SceneChangeScript;
```

(continues on next page)

(continued from previous page)

```
        if (sceneChangeObject != null) {
            // Yes, get our current scene index
            int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
            // Load up the scene accourding to the sceneChange value
            UnityEngine.SceneManagement.SceneManager.LoadScene(currentSceneIndex +␣
↪sceneChangeObject.sceneChange);
        }
    }
    void OnGUI()
    {
        // Dispaly our score
        GUIStyle guiStyle = new GUIStyle(GUI.skin.label);
        guiStyle.fontSize = 32; //modify the font height
        GUI.Label(new Rect(10, 10, 250, 50), "Score: " + score, guiStyle);
    }
}
```

CHAPTER

EIGHT

2D SHOOTING

Contents

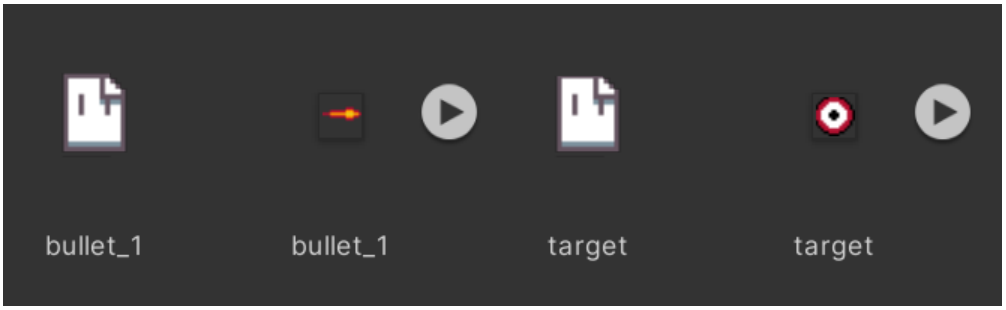
There are three main ways to shoot things in Unity.

- For laser-types of things where you insta-hit, you can use **ray-casting**.
- You can move sprites and check by distance.
- You can move sprites, and use colliders. This is how we are going to demo things here.

8.1 Make a sprites in Aseprite

We need a target and a projectile.

- Make a bullet, laser, heart, whatever projectile you want to shoot.
- Make a target to hit.
- Export, and import into Unity changing the normal three things. (pixels per unit, compression, filter)



8.2 Detect mouse down events

Now, we will use the mouse button to shoot. First, we need to detect mouse-down events.

In our `Update` method on cour controller (not `FixedUpdate`, doesn't seem to work well), we can detect a mouse-down event with `Input.GetMouseButtonDown(0)`. The 0 is for our left mouse button. An implementation might look like:

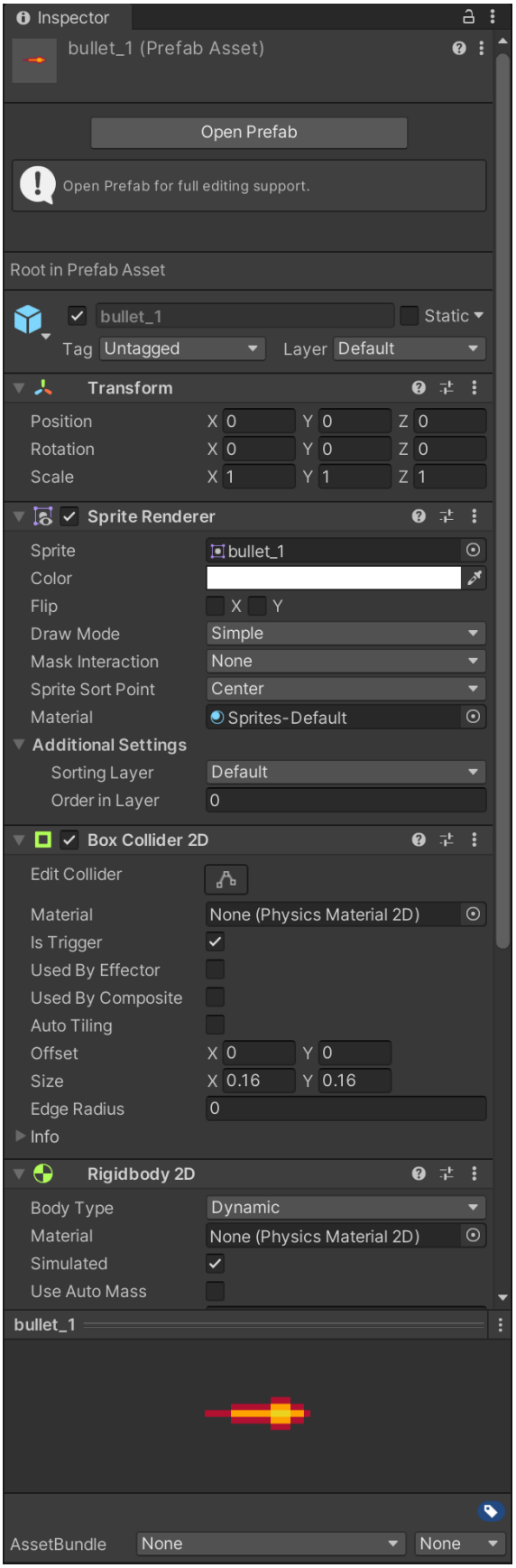
```
// Has the mouse been pressed?
if (Input.GetMouseButtonDown(0))
{
    Debug.Log("Mouse down");
}
```

Code and confirm it works.

8.3 Create a bullet

Now we need something to shoot.

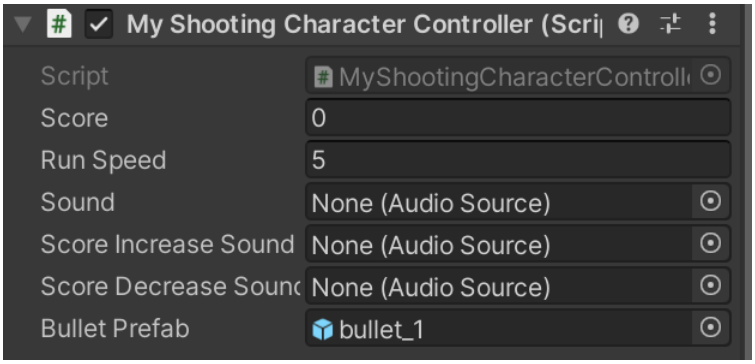
- Create a bullet prefab.
- Add a box collider so we can detect collisions. Set the collider to be a trigger, as we don't want it bumping into things.
- Add a rigidbody so we can move it via physics.


Inspector
bullet_1 (Prefab Asset)
Open Prefab
Open Prefab for full editing support.
Root in Prefab Asset
bullet_1
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite bullet_1
Color
Flip X Y
Draw Mode Simple
Mask Interaction None
Sprite Sort Point Center
Material Sprites-Default
Additional Settings
Sorting Layer Default
Order in Layer 0
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X 0 Y 0
Size X 0.16 Y 0.16
Edge Radius 0
Info
Rigidbody 2D
Body Type Dynamic
Material None (Physics Material 2D)
Simulated
Use Auto Mass
bullet_1
AssetBundle None None

Go to your character controller cand add a public variable for the prefab. Code would look like:

```
public GameObject bulletPrefab;
```

Then drag the prefab into the new blank spot in your character.



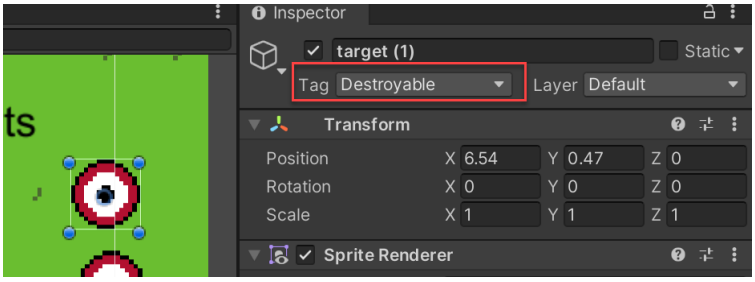
Update code to fire the bullet:

```
// Mouse pressed?
if (Input.GetMouseButtonDown(0))
{
    // Make a bullet
    var bullet = Instantiate(bulletPrefab, body.position, Quaternion.identity);
    // Get the body of the bullet
    var bulletbody = bullet.GetComponent<Rigidbody2D>();
    // Move the bullet to the right
    bulletbody.velocity = new Vector2(4, 0);
}
```

It would be better code if you make the speed a public variable rather than hard-code it. And we'll get to aiming in a bit.

8.4 Create targets

Now we need something to shoot. Create targets. Add a collider. Add a tag for "Destroyable".



8.5 Add a bullet script to destroy

This bullet script will destroy itself after moving 8 units, or it will destroy an object tagged 'destroyable'.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BulletScript : MonoBehaviour
{
    Vector3 _origin;
    public float maxDistance = 8.0f;

    // Start is called before the first frame update
    void Start()
    {
        // Get position we started at, so we can see how far the bullet traveled.
        _origin = transform.position;
    }


    public void OnTriggerEnter2D(Collider2D collision)
    {
        Debug.Log("Trigger");
        if (collision.tag == "Destroyable")
        {

            Debug.Log("Destroyable");
            // Destroy item we hit
            Destroy(collision.gameObject);
            // Cause bullet to destroy itself
            // Put this outside the if to get deleted when hitting non-destroyable␣
↪objects
            Destroy(gameObject);
        }
    }

    // Update is called once per frame
    void Update()
    {
        // How far has the bullet gone?
        float distance = Vector2.Distance(_origin, transform.position);
        // If too far, then remove ourselves from the game.
        if (distance > maxDistance)
        {
            // Cause bullet to destroy itself
            Destroy(gameObject);
        }
    }
}
```

8.6 Calculate angles

Next, if we want to fire in a particular direction, we need to do some math. Here's the code with comments.

```
// Get the angle of a vector
public float GetYRotFromVec(Vector2 v1)
{
    float _r = Mathf.Atan2(v1.y, v1.x);
    float _d = (_r / Mathf.PI) * 180;

    return _d;
}

void Update()
{
    // Get our axis values
    horizontal = Input.GetAxisRaw("Horizontal");
    vertical = Input.GetAxisRaw("Vertical");

    // Has the mouse been pressed?
    if (Input.GetMouseButtonDown(0))
    {
        // -- Fire a bullet

        // Create the bullet
        var bullet = Instantiate(bulletPrefab, body.position, Quaternion.identity);
        // Get a reference to the bullet's rigid body
        var bulletbody = bullet.GetComponent<Rigidbody2D>();
        // Where is the mouse on the screen?
        var mousePosition = Input.mousePosition;
        // Where is the mouse in the world?
        Vector3 target3 = Camera.main.ScreenToWorldPoint(mousePosition);
        // Set the z value of this vector 3
        target3.z = 0;
        // What is the normalized vector from the player to the mouse?
        Vector2 direction = (target3 - transform.position).normalized;
        // What is the angle in degrees?
        float angle = GetYRotFromVec(direction);
        // Rotate the bullet
        bulletbody.rotation = angle;
        // Give the bullet speed
        bulletbody.velocity = direction * bulletSpeed;
    }
}
```

CHAPTER

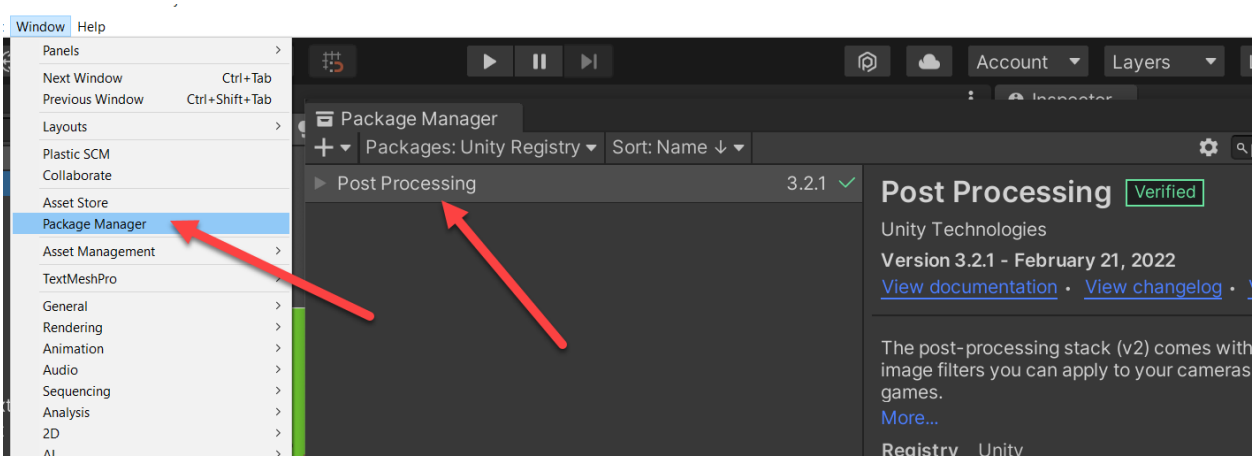
NINE

ADDING A BLOOM EFFECT

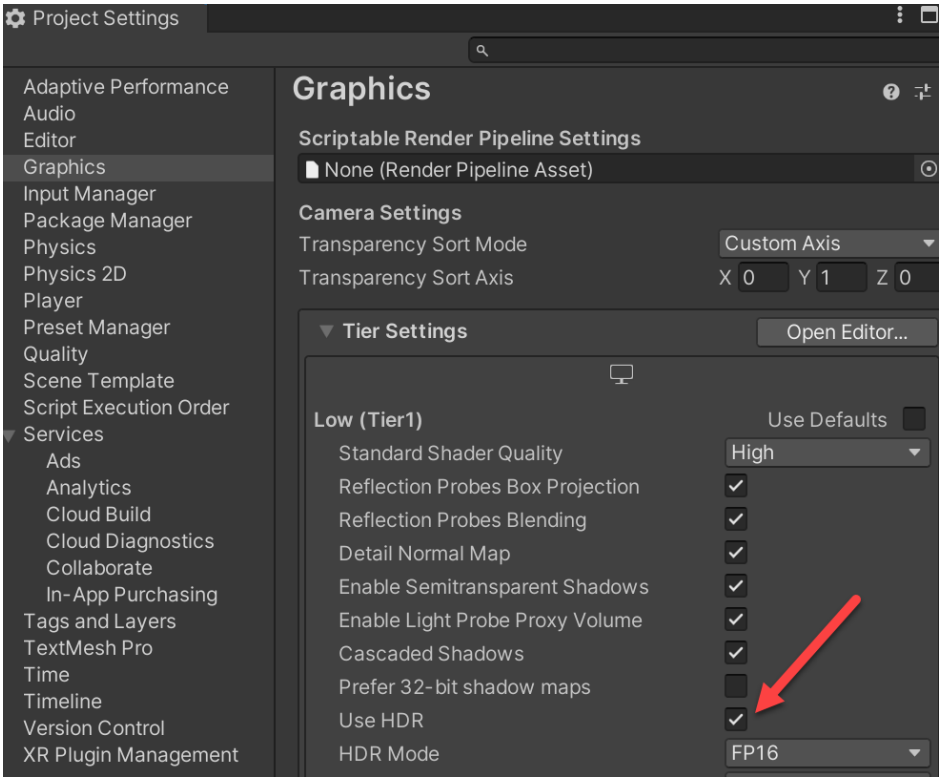
Contents

9.1 Step 1 - Add the post processing package to your project

Go to Window -> Package Manager and then install the "Post Processing" package. This is project-wide so this only needs to happen once for an entire project.

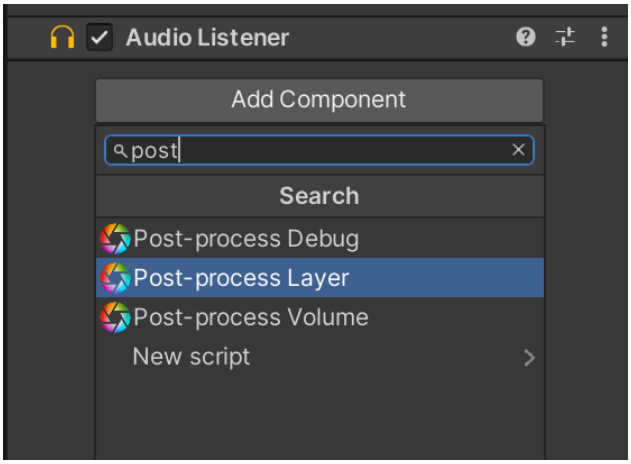


9.2 Step 2 - Enable HDR for the project

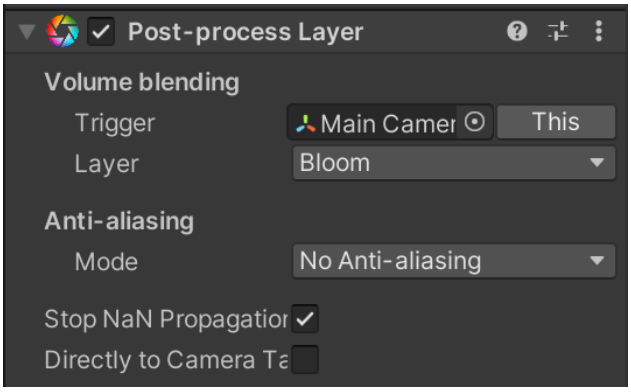


9.3 Step 3 - Add a post-processing layer to the camera

Select the camera. Add a post process layer component to the camera.



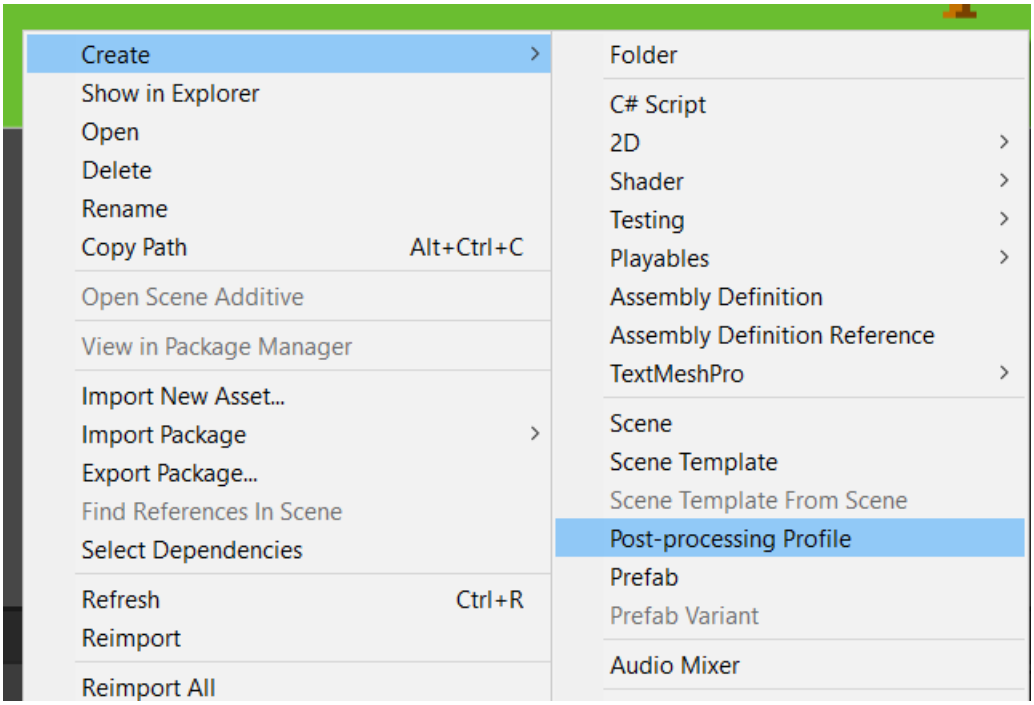
Select the 'Bloom' layer. You may need to create this layer if it does not yet exist for your project.



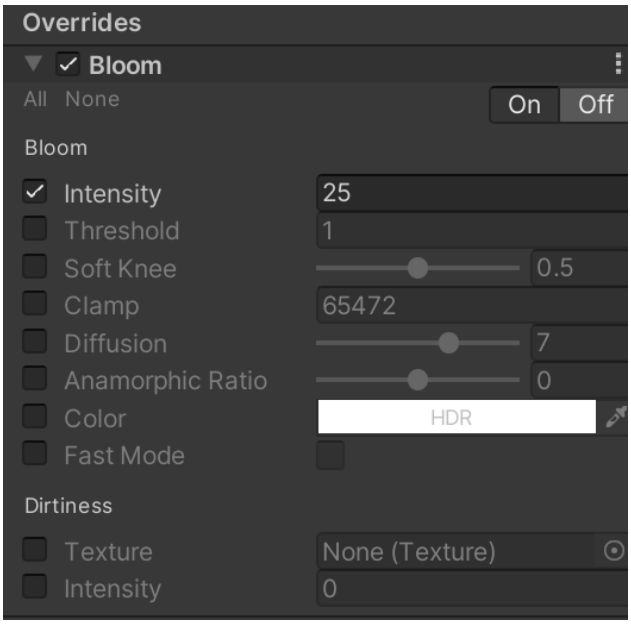
9.4 Step 4 - Create a post processing profile

Find/create a directory for post processors.

Create a post processor:



Add a bloom effect:

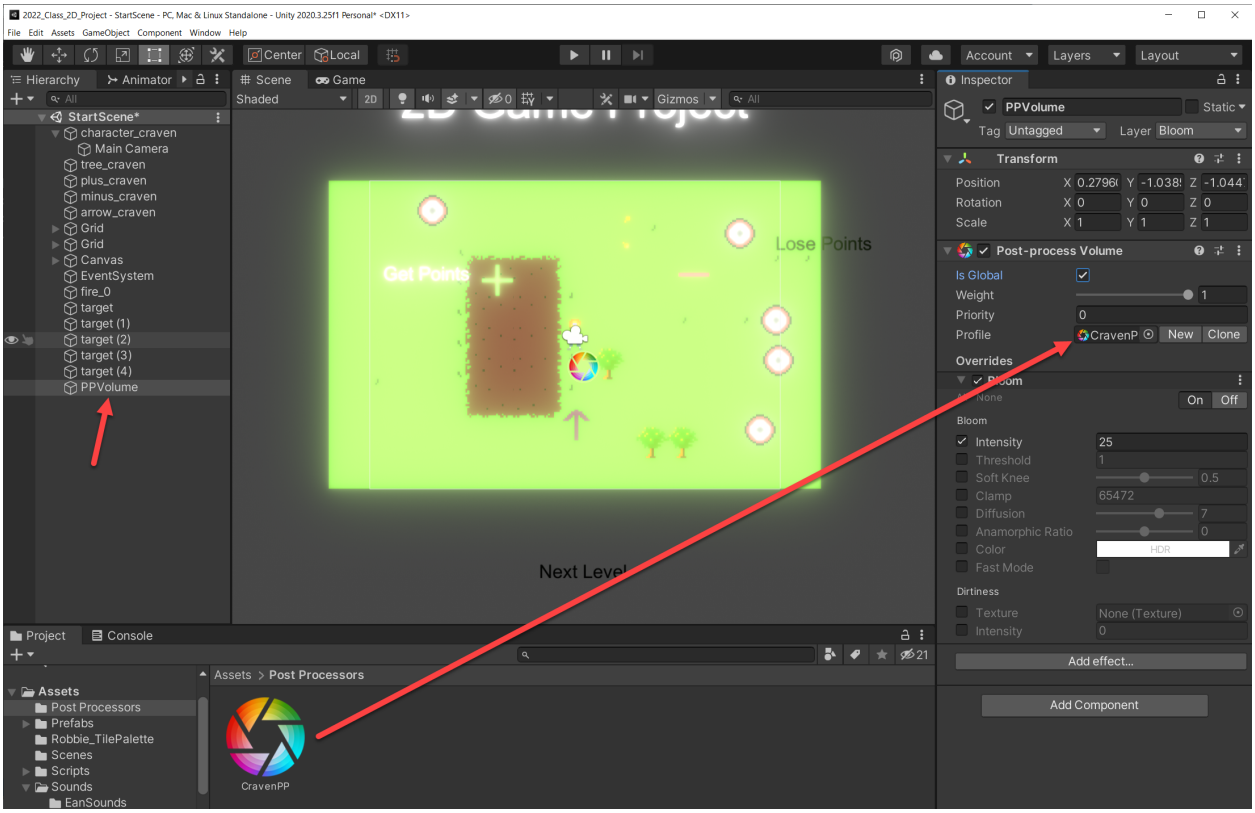


9.5 Step 5 - Create a post processing volume

Go to your project, add an empty. Call it “post-process bloom” or something like that.

Add a “Process Volume” component to it.

Drag in the post processor to the proper field.



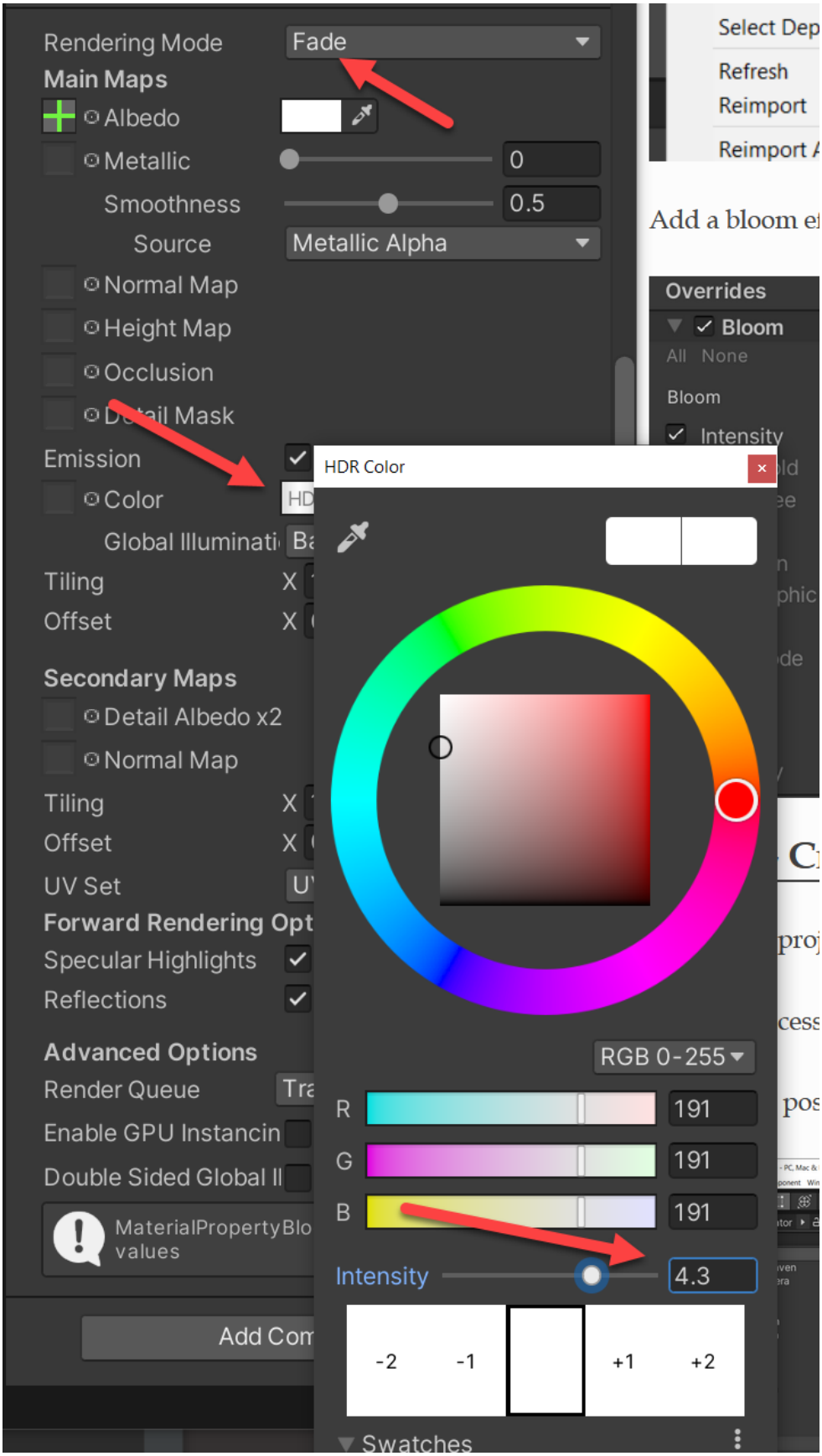
This makes everything glow, fine if you are doing some neon geometry wars thing. But what about just one thing?

9.6 Step 6 - Make one thing glow

Set post-processing intensity to 1. Zero turns it off, we don't want that. Above 1 will make everything glow. Don't want that.

Create a new material called "Glow".

Give it the following properties:


Rendering Mode
Fade
Main Maps
Albedo
Metallic
0
Smoothness
0.5
Source
Metallic Alpha
Normal Map
Height Map
Occlusion
Emission
Color
Global Illuminati
Tiling
Offset
Secondary Maps
Detail Albedo x2
Normal Map
Tiling
Offset
UV Set
Forward Rendering Opt
Specular Highlights
Reflections
Advanced Options
Render Queue
Enable GPU Instancin
Double Sided Global Il
values
Add Com
HDR Color
RGB 0-255
R
191
G
191
B
191
Intensity
4.3
-2
-1
+1
+2
Swatches
Select Dep
Refresh
Reimport
Add a bloom e
Overrides
Bloom
All None
Bloom
Intensity

You have to specify the color, it doesn't pick it up from the image.

CHAPTER

TEN

2D PARTICLE SYSTEM

Contents

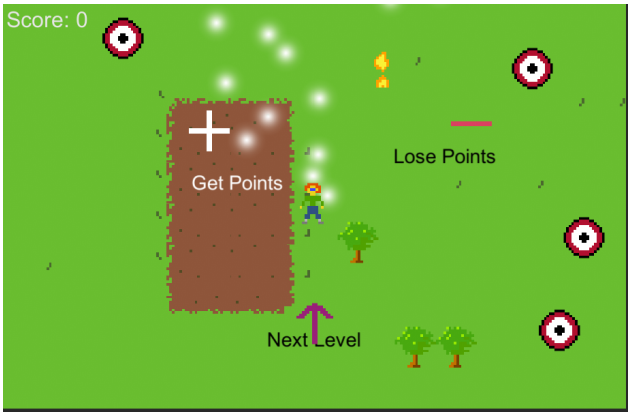
Let's make particles! For a YouTube video that covers this, see: https://www.youtube.com/watch?v=_z68_OoC_0o

10.1 Create a white sprite particle

Use Aseprite

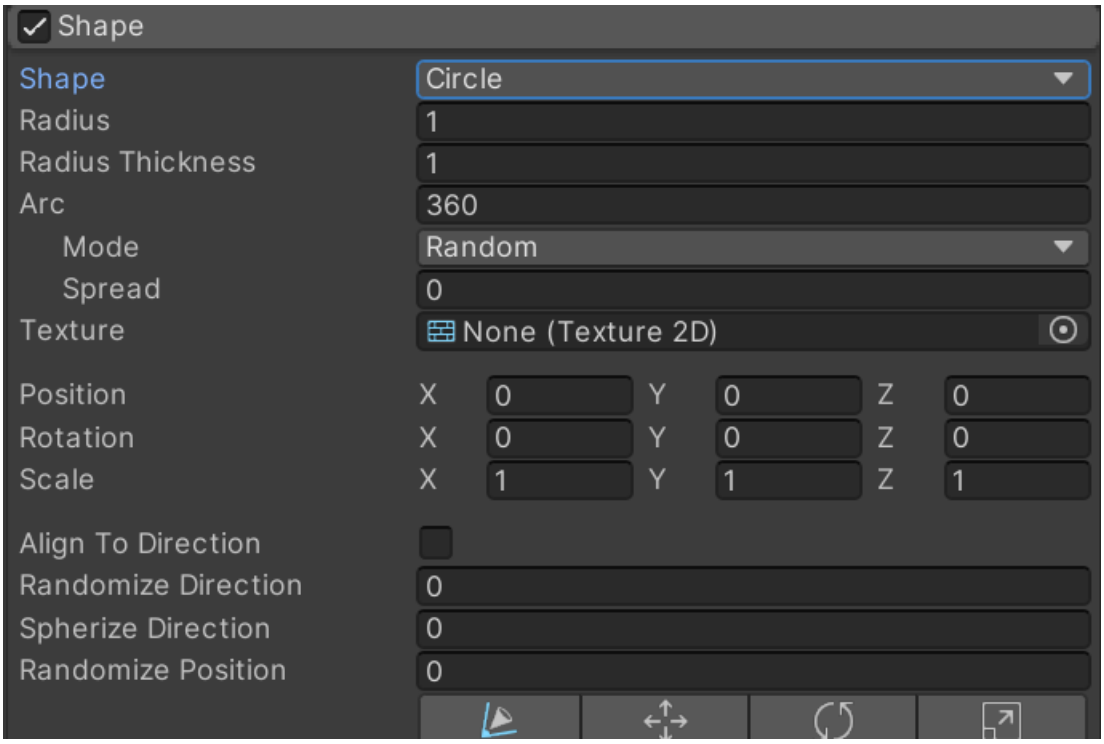
10.2 Add a particle system

In Unity, select GameObject -> Effects -> Particle System. You should now have a new particle system in your game throwing off fuzzy dots.



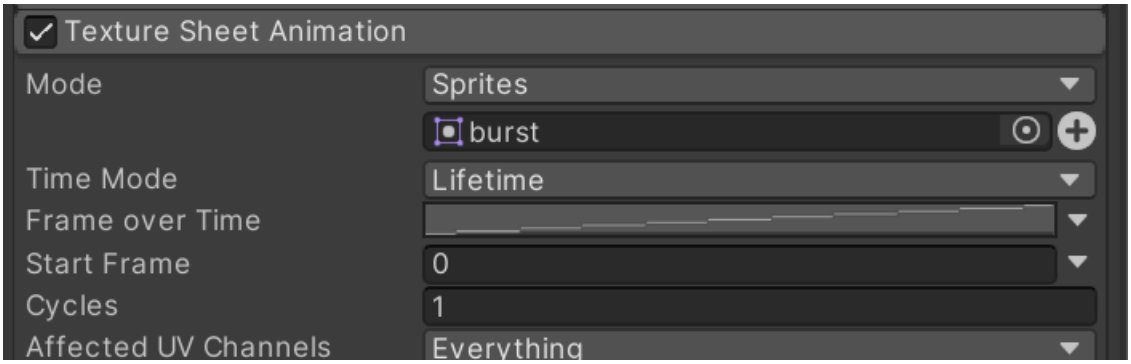
The rotation of the default system has the particles flying up. Take out the -90 rotation on the particle game object and the particles fly towards the camera. Experiment with it.

Experiment with shape of emitter.

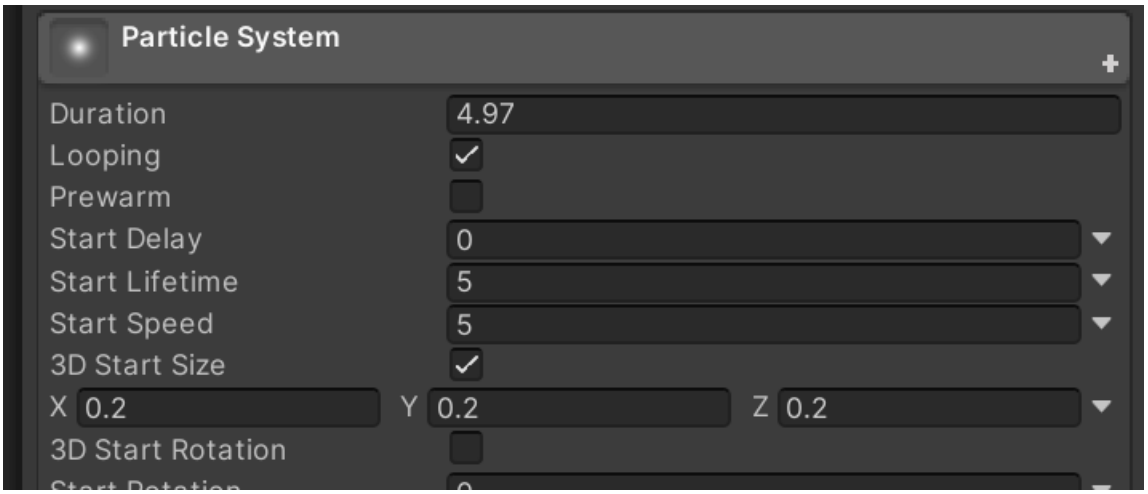


Add gravity to make the particles fly down.

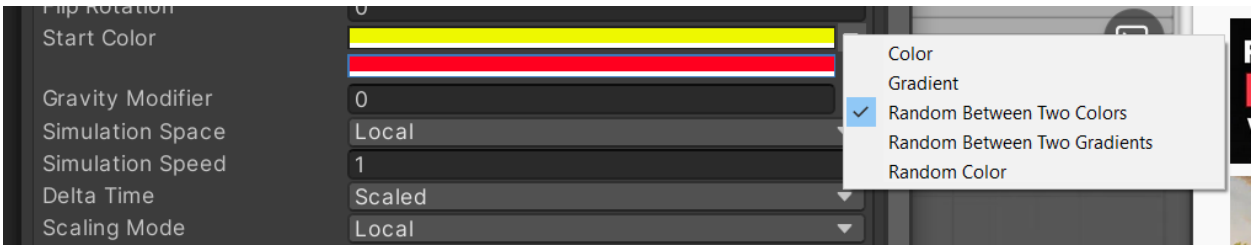
10.3 Make the particles sprites



10.3.1 Scale the particles



10.3.2 Color the particles



10.3.3 Amount of particles

Adjust "rate over time"

10.3.4 Particle trails

Try adding trails, as shown in the video.

10.4 Make things blow up when hit

Update your code so that your bullet script will create a "burst" prefab when you hit an item. You'll need to have the prefab be created with a script that will destroy itself over time.

Note: This example just shows the important parts. It doesn't show the needed "make the bullet disappear after a while." We showed that earlier. You'll need to combine your scripts.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BurstBulletScript : MonoBehaviour
{
    public GameObject burstPrefab;
    Rigidbody2D body;

    // Start is called before the first frame update
    void Start()
    {
        body = GetComponent<Rigidbody2D>();
    }


    public void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Destroyable")
        {
            // Destroy the item
            Destroy(collision.gameObject);
            // Create the 'burst' effect
            var burst = Instantiate(burstPrefab, body.position, Quaternion.identity);
        }
    }
}
```

CHAPTER

ELEVEN

2D ATTACKS

This section based off: https://www.youtube.com/watch?v=1QfxdUpVh5I

11.1 Add time-limited trigger for attacks

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CravenAttackScript : MonoBehaviour
{
    // How frequently can we attack?
    public float attackTimeLimit = 0.5f;

    // Countdown timer for attacks
    private float attackCountdownTimer = 0;

    void Update()
    {
        // See if we can attack, via timer.
        if (attackCountdownTimer <= 0)
        {
            // We can attack. See if user hit space bar.
            if (Input.GetKey(KeyCode.Space))
            {
                Debug.Log("Attack");
                attackCountdownTimer = attackTimeLimit;
            }
        }
        else
        {
            // Attack timer needs count-down
            attackCountdownTimer -= Time.deltaTime;
        }
    }
}
```

11.2 Do damage

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CravenAttackScript : MonoBehaviour
{
    // How frequently can we attack?
    public float attackTimeLimit = 0.5f;

    // Countdown timer for attacks
    private float attackCountdownTimer = 0;

    // An empty parented that says where to attack
    public Transform attackPos;
    // Radius of attack circle
    public float attackRange;
    // What layer will the enemies be on?
    public LayerMask enemyLayer;
    // How much damage to deal
    public int damage = 3;

    void Update()
    {
        // See if we can attack, via timer.
        if (attackCountdownTimer <= 0)
        {
            // We can attack. See if user hit space bar.
            if (Input.GetKey(KeyCode.Space))
            {
                Debug.Log("Attack");
                // Reset the countdown timer
                attackCountdownTimer = attackTimeLimit;
                // What enemies did we hit?
                Collider2D[] enemiesToDamage = Physics2D.OverlapCircleAll(attackPos.
↪position, attackRange, enemyLayer);
                // Loop through each enemy we hit
                for(int i=0; i < enemiesToDamage.Length; i++)
                {
                    // Get the enemy script attached to this object
                    CravenEnemyScript enemyScript = enemiesToDamage[i].GetComponent
↪<CravenEnemyScript>();
                    // If there is an enemy script
                    if (enemyScript)
                    {
                        // Damage
                        enemiesToDamage[i].GetComponent<CravenEnemyScript>().health -=␣
↪damage;
                        // Print health levels
                        Debug.Log(enemiesToDamage[i].GetComponent<CravenEnemyScript>().
↪health);
```

(continues on next page)

(continued from previous page)

```
                        // --- ToDo: destroy enemy here when health <= 0
                    }
                    else
                    {
                        // We hit an enemy, but there's no script attached to it.
                        Debug.Log("Enemy Script not present");
                    }
                }
            }
        }
        else
        {
            // Attack timer needs count-down
            attackCountdownTimer -= Time.deltaTime;
        }
    }
    // Used to draw a circle when we are selecting the player in the scene view
    void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(attackPos.position, attackRange);
    }
}
```

Note: You'll need: * An enemy script * Turn on gizmos in the scene view * An enemy layer * Program a change to the attackPos when user changes direction.

CHAPTER

TWELVE

CAMEL IN C#

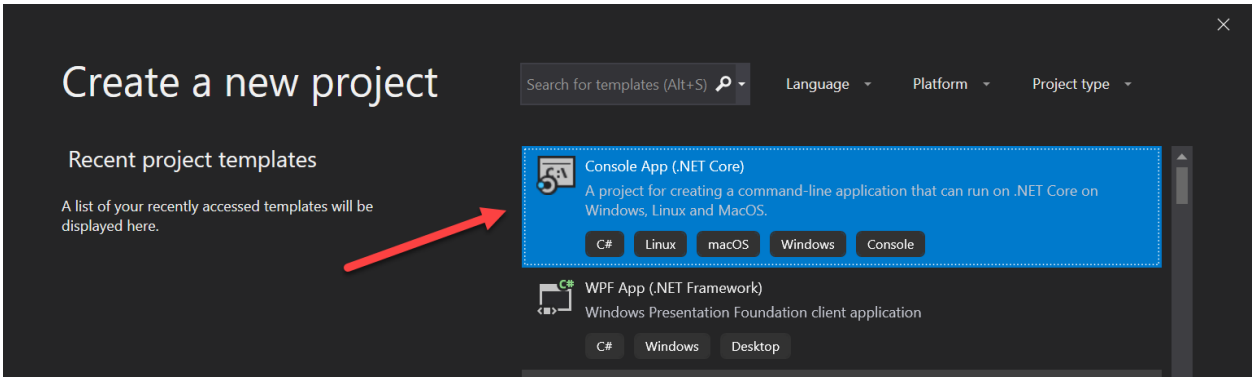
If you had me for CMSC 150, you likely remember the Camel game. Your task for this assignment is to code the Camel game in C#.

Here is the link for the description of the Camel game:

https://arcade-book.readthedocs.io/en/latest/labs/lab_04_camel/camel.html

Use Visual Studio. It is free to install. You can download it from here: https://visualstudio.microsoft.com/downloads/

Create a new console app project, and call it `Camel`:



This open with a "Hello World" program. Run the program. It will appear in a separate console window as opposed to a window in the IDE.

Here's some code to get started:

```
using System;

namespace Camel
{
    class Program
    {
        static void Main(string[] args)
        {
            // Introductory message
            Console.WriteLine("Welcome to Camel!");

            // Main game loop
            bool done = false;
            while (!done)
            {
```

(continues on next page)

(continued from previous page)

```
                // Print commands
                Console.WriteLine();
                Console.WriteLine("A. Drink from your canteen.");
                Console.WriteLine("B. Ahead moderate speed.");
                Console.WriteLine("C. Ahead full speed.");
                Console.WriteLine("D. Stop and rest.");
                Console.WriteLine("E. Status check.");
                Console.WriteLine("Q. Quit.");

                // Get user command
                Console.Write("What is your command? ");
                string userCommand = Console.ReadLine();
                Console.WriteLine();

                // Process user command
                if (userCommand == "a")
                {
                    Console.WriteLine("You drank from the canteen.");
                } else
                {
                    Console.WriteLine("Unknown command.");
                }
            }
        }
    }
}
```

Part of this task is practicing how to quickly search up answers. I'm not going to step through how to code in C#, you have enough talent to get started on your own.

We will review some of programs together so we can get ideas from each other.

Today, make sure you have created a project that can print "Hello World." By the time you come to class Thursday, have a start to the main game loop.

While it is possible to code the program in one function and loop, see if you can use good design and break the parts into functions.

Feel free to change the theme and add features.

If you change the theme, you must still have a number line you are traveling across, some kind of resource you can run out of, and "something" that can catch you.

Be ready to present your work on Thursday and your final project on Tuesday.

To turn in, upload GitHub URL to your project.

CHAPTER

THIRTEEN

ROLL-A-BALL

- Follow the Unity tutorial for roll-a-ball: https://learn.unity.com/project/roll-a-ball
- Put into git, and use this .gitignore file: https://github.com/github/gitignore/blob/main/Unity.gitignore
- Upload to GitHub
- Create a readme and include a screenshot
- Tuesday next week will be workday
  - Bring in something mostly working.
  - Get help with github
  - Get help with readme and image
- Thursday we'll demo our games
- Turn in github url

CHAPTER

FOURTEEN

CUSTOM ROLL-A-BALL

- Start with your prior roll-a-ball assignment.
- Create at least two objects in Blender, and add them as obstacles in your game.
  - Obstacles must be at least as complex as the trees we created.
  - Obstacles must not be the same trees we created. Do something different.
  - Obstacles must have at least two different materials on them.
- Update the collectable to have a custom shape you create in blender.
  - Feel free to turn off the rotation if it doesn't work for your collectable.
- Update your playing field with a custom playing field created in Blender.
- Grading will be a somewhat subjective. Impress me, don't shoot for the minimum.
- Update the image in your read-me.
- Turn in a Git-Hub link to your project.

CHAPTER

FIFTEEN

TEAM 3D GAME WORK

For this project (and the coming weeks) you'll break into groups and develop a 3D game. Your goal is to improve this game, week-by-week.

15.1 Starting the Project

- Start with one of your roll-a-ball games as a working base.
- Invite other team members to that project.
- You must cite any 3d models, textures, sounds that you use from another source.
  - Get this started by creating a section in your readme to hold this info.
- Figure out how you want to stay in contact. Exchange info. (E-mail, slack, discord, etc.)
- Brainstorm Ideas
  - Theme?
  - Color scheme? See Adobe Kuler and create a swatch?
  - Create one or more tasks for everyone. Backup tasks are a good idea. See "Some Ideas On Tasks"

15.2 Each Week on Thursday

- Pick goals/tasks for each person to get done that week.
- Enter each task as an issue in GitHub.
- Assign it to the proper person.
- If you notice bugs, enter them in GitHub and assign to the proper person.

15.3 Each Week on Tuesday

- Make sure everyone's work has been merged into GitHub.
- Test your application to make sure it is working.
- Help each other with the tasks and bugs.

15.4 Some Ideas on Tasks

- Object modeling
  - Create an interesting playing field
  - Add more objects, add more detail to objects
- Materials
  - Learn some of the options for materials, and make things shiny, etc.
  - Note: Learn these in Unity, not Blender. Blender to Unity material transfer is limited.
- Textures
  - Map an image onto an item (UV Mapping) Can do in Blender or Unity.
  - Learn to use normal maps
  - Create water
- Shaders
  - Work with shaders to create better looking materials
- Lighting
  - Instead of one generic light, add better lighting. Spot lights, lamps, etc.
- Skybox
  - Learn to add a skybox. **Warning:** Be very careful about how hi-res of an image you download. These can be huge and blow up your project if you download something too big.
- Sound
  - Add sound for pickups
  - Add sound when bouncing into objects
  - Add sound for movement
- Create level system
  - Go to new level if all items are picked up
  - Go to new level if player gets to a goal point
- Enemies
  - Create items that reset the user to the start if you bump into them
  - Create have player lose a life when hitting enemy
  - Support 'game over' when player loses all lives
  - Have enemy move towards player

  - Investigate path finding to have enemy move around objects
- Particles
  - Create liquids, smoke, clouds, flames, magic effects
- Shooting
  - Be able to shoot things. Enemies, collectables, walls.
- UI
  - Create intro/instruction screens
  - Allow game restart
  - Show lives left
  - Add background/panel to UI
  - Add dialog system (encounter NPC, have popup dialog)
- Multiplayer
  - Add networking
- Animation
  - Animate obstacles
  - Make moving platforms
  - Create switches that trigger events
  - Create a 3d car instead of a ball to move around
  - Create a 3d walking character rather than a rolling ball. Use Mixamo.
- Player
  - Add ability to jump
  - Add ability to run

15.5 Important Notes

- Do not add assets into a folder without using Unity. This will lead to merge errors that will lose you a lot of time.
- If working on a challenging item, have a back-up goal. You've got to get something done, so you don't want to be stuck if things are more complex than expected.
- Everyone must be on the same version of Unity. Do not upgrade your Unity. That will force everyone to upgrade, or you'll just end up losing your work.
- Your work must be integrated. For example, if your task is designing a tree, don't spend all your time making a beautiful tree in Blender and never get it into the game. Create a cylinder in Blender. Get it into the game. Fancy it up with some branches. Get that in the game. Add materials. Get that into the game. If something isn't in the game, it might as well not exist.
- Commit early. Commit often. If you only commit during one day this week, it won't look like you've done much work at all.
- The fancy materials and modifiers you use in Blender are not likely to show up in Unity. Keep it simple. Make sure things work in Unity before sinking a lot of time into them.

15.6 Turn In

Turn in a report.

- Summarize what you finished this week.
- Link to the GitHub project.
- Link to the issue that has the item(s) you worked on.
- Link to your commits. It will look something like: https://github.com/pythonarcade/arcade/commits?author=pvcraven
- Include an image of what you did, and show it working in the game.

15.7 Grading

I'll grade the way I evaluated the work of my employees back when I worked IT.

- Integration with the project. When I hit 'play' on the game, can I see what you did? If so, that will help give you a good grade. Don't make the mistake of adding a model, sound, material, or some other component, but not make it part of gameplay. If I hit 'play' and can't see your work, then it serves no purpose. When adding items, start with a simple version. For example, a cube, a beep, code that just prints "hello world" at the right trigger. You have something working. Go back and add detail. Always keep it in the playable game.
- Frequency of commits. Do you have commits spread across three or more days? This shows ongoing work and integration with the whole project. In the workplace, I'd expect commits every day. Or hour or two. If you are doing something that might break the project, do it in a separate branch, then merge. Ask if you'd like help learning to do this.
- Quantity/complexity of work. Did you do some scripting? Or add a detailed model? Or add a lot of different low-poly models?
- Documentation. Did you include links to your project and your commits? Did you detail what you did that works in words? Include screenshots? Did you make it so simple to see what you did, I don't even need to clone the game? Did you see me in class and show off your work there? Did you use the issue tracking? As a manager, I'm looking at that more than diving into your code. You don't want managers diving into the code, make it easy for them to track progress.
- Citations.

CHAPTER

SIXTEEN

2D ASSIGNMENT 1

In this assignment, we'll get started with 2D.

16.1 Requirements

Turn in a report detailing and showing (with screenshots) your completion of:

Table 1: Point Allocation

Item	Points
Sprite 1	10
Sprite 2	10
Sprite 3	10
Sprite 4	10
Sprite 5	10
Scene	10
Proper collision	10
Implement scored items	10
Implement next level transition	10
Get something other than player moving	10

Scoring:

- 0 pts - not implemented
- 1-5 pts - buggy
- 6-7 pts - meets minimum requirements. i.e., it works.
- 8-9 pts - Expanded beyond minimum requirements
- 10 pts - Expanded into something that looks like an actual game.

16.2 Directions

To get started, clone our project. Get invited as a collaborator.

https://github.com/pvcraven/2022_Class_2D_Project

All item names must include yours so we can identify them.

Then most of what you need to get started is at: *2D Unity Part 1*.

The main thing not covered is getting some objects to move via scripts. I'm leaving that up to you to figure out.

16.3 Sample Items to Create

- Outdoors
  - Tree
  - Rock
  - Fence
  - Grass
  - Flowers
- House or some building
- Icons
  - Pencil
  - Hand
- Hand-held Items
  - Wand
  - Sword
  - Staff
- Gems
- Coin
- Potion
- Clothing
  - Boot
  - Shoe
  - Shirt
  - Vest
  - Hat
  - Helmet
- Food
  - Fruit
    - Pumpkin

    * Apple
    * Pear
    * Orange
    * Grapes
    * Pineapple
    * Raspberry
    * Watermelon
    * Strawberry
    * Cherries
    * Bananas
- Other food
    * Mushroom
    * Ice cream code
    * Donut
    * Cookie
    * Pizza

CHAPTER

SEVENTEEN

2D ASSIGNMENT 2

In this assignment, we'll get continue work on our 2D level. This assignment will concentrate on:

- 2D Tile maps
- Adding sound effects

17.1 Requirements

Turn in a report detailing and showing (with screenshots) your completion of:

Table 1: Point Allocation

Item	Points	Scoring
Basic Tiles Created	15	Set of 12 background tiles, as shown in class. Must have some detail to get full 15 points.
Additional tiles created	15	Create at least three additional tiles. I'd suggest something that would be an obstruction, or additional background tiles.
Background tile map layer	20	Background tile layer. Must be reasonably extensive for the full 20 points.
Collision tile map layer	15	Add things to run into as part of tile tile map.
Sound effects added	20	Sound effect for points up, points down, and level up/down.
Unity background music	15	Play some background music. Don't forget to turn it off when changing levels.

17.2 Directions

Most of what you need to get started with the tiles can be found in *2D Unity Part 2*.

The sound effects and background music should not be too hard. I'm leaving it up to your web-search skills to figure out how to add that.

CHAPTER

EIGHTEEN

2D ANIMATION ASSIGNMENT

Like earlier assignments, please create a doc that points to the work that you did. Get practice documenting and showing off your work, as this is going to be very important for getting promoted in your career.

Table 1: Point Allocation

Item	Points	Scoring
Create animated time-based sprite frames 1	10	Create at least 8 frames of a time-based sprite. This can be from the fire or water sprite created in class during *2D Animation*. To turn in, take a screenshot and show the resulting images.
Create animated time-based sprite in Unity 1	5	Get the animated sprite from above working in Unity. Slow down the keyframes so it doesn't run full speed.
Create animated time-based sprite frames 2	10	Create at least 8 frames of a time-based sprite.
Create animated time-based sprite in Unity 2	5	Get the sprite working.
Create animated time-based character sprite frames	10	Create "idle" and "walking" animations in Aseprite. Show the images created.
Create animated time-based character sprite in Unity	10	Idle animation must play. When character moves, the walking animation must start. When character is done walking, go back to idle. Select left/right based on direction character is moving.
Create animated time-based NPC sprite frames	10	Create one more moving animation, this time for an NPC. Must have two different animations. (Idle/walk)
Create animated time-based NPC sprite in Unity	10	Get both animations and transition to work for NPC character.
Expand level	10	Take the level that you have, and make it bigger. I suggest showing the original level, and the expansion. You'll probably want at least two screens worth of additional layout.
Total	100	

CHAPTER

NINETEEN

2D FINAL ASSIGNMENT

Like earlier assignments, please create a doc that points to the work that you did. Get practice documenting and showing off your work, as this is going to be very important for getting promoted in your career.

Table 1: Point Allocation

Item	Points	Scoring
Add the ability to fire projectiles	20	Some kind of projectile needs to be fired.
Create a destroyable target	15	Have an item that is destroyed when hit by the projectile
Create something that glows	15	At least one item needs a glow effect
Create a particle system	20	For full points, the particle effect needs to be triggered. For example, an explosion when an item is hit. Stand-alone particle systems that just emit are worth about 16 points. Scoring will be somewhat dependent on amount of particle features used.
Create an 'attack' system.	20	?
Demo your level.	10	Be in class for our last class, and show off your level.
Total	100	