The Photographer's Guide to Los Santos

Contents

5 Nature Documentary 15

6 Surrealist Photography 23

7 Hyperrealist Photography 37

Preface

"Los Santos. The city of shitheads. Where else would he be?"

— Trevor Philips

Chapter 1

Introduction

About The Photographer's Guide to Los Santos

The Photographer's Guide to Los Santos sits between a touristic guide and a photography manual, and between an exhibition catalogue and a peak behind the scenes of artwork creation.

The Photographer's Guide to Los Santos is an ongoing project that builds on top of a research on artistic practices within spaces of computer games, with a particular focus on in-game photography, machinima and digital visual arts. It follows some themes and ideas previously explored in the exhibition *How to Win at Photography*, while focusing more specifically on the relationship between computer games and photographic activities inside the world of Grand Theft Auto V.

The idea of a guide refers to in-game photography as a form of 'virtual tourism' (Book, 2003), which was also the premise of an actual tourist guide published by Rough Guides in their 2019 *Rough Guide to XBOX*. Yet this guide project also understands the game world as a site for image production and artistic creation, turning the game into a destination for a 'game art tourist'. The Photographer's Guide to Los Santos presents the game environment of Grand Theft Auto V both as a space to explore and in which to create images, as well as a place to navigate and learn about some of the most important artworks that it has enabled to create.

The project also brings together several experiences from teaching in-game photography as an artistic practice in different educational settings and institutions, compiling materials and tools for students and artists interested in engaging with the field. The tourist guide of the game world doubles as a photography manual for the in-game photography age, featuring tutorials and excercises ranging from game screenshotting to computer programming for creative modding. Through

the practical excercises, the project invites to rethink the game object as a space for creative, subversive and critical endeavours, which can be played differently, documented, reclaimed or modified through an artistic approach.

Finally, the project draws inspiration from the works of artists who have explored the 'metaplay' of photographing game words instead of following the game rules and attempt to reach the goal of winning. The Photographer's Guide to Los Santos is indebted to all the artists it features, but was particularly inspired by Gareth Damian Martin's live streamed workshop *Photography Tour of No Man's Sky* (realized for Now Play This Festival 2020), Total Refusal & Ismaël Joffroy Chandoutis's 2021 in-game lecture performance and guided tour *Everyday Daylight* (realized for the CCS Paris), and Alan Butler's epic 2020 live endurance performance *Witness to a Changing West* (realized for Screen Walks) and his 'Content Replication Assignments'.

Grand Theft Auto V Studies

Los Santos is the Grand Theft Auto V's fictional, parodic version of real-life Los Angeles. Just like Los Angeles is the global centre of film and commercial media production, Los Santos is the epicentre of in-game photography and machinima creation. While it may seem reductive to only focus on a single game to address the larger phenomenon of in-game photography, GTA V is the biggest source of creative outputs to date, with its extended open world and one of the largest community of active modders.

Launched in 2013, the game contains a world map of more than 80 square kilometers of total area, which includes the urban area of the city of Los Santos and the rural area of Blaine County. This incredibly vast environment features a large desert region, dense forest, several mountains, beachside towns, on top of the large metropolis of Los Santos. The game simulates the everyday life of hundreds of individual NPCs (while it allegedly counts a population of over 4 million) as well as counting 28 animal species, and more than 800 buildings in GTA V are based on real-life landmarks.

The size of the photorealistic simulation is only matched by the complexity of the game engine and its code, which - thanks to the effort of GTA V's modding community - allows players to use the game world as a powerful tool to create new scenes, take controls of its algorithmic entities, modify cameras and reshaping the game into a movie set or a photo studio.

Grand Theft Auto V Tourism

The project can be seen as something between a playful travel guide of Los Santos and one of the star maps offered to tourist in Hollywood, pointing to the

homes of movie actors and hollywood celebrities. This guide allows players to explore the game environment following some of the most interesting artworks that have been created with(in) it. It's divided in thematic chapters that follow different artistic practices, taking place in different locations of the game environment, followed by different tutorials and exercises connected with the works and the space analyzed.

The themes explore different approaches and practices connected to established artistic and photographic currents, with a general introduction text that gives an overview of the ideas, contexts and issues connected to the specific topic. A selection of artworks for each themes is presented by a curatorial statement, introducing the work and its artistic relevance. The work is accompanied by information on the in-game location in which it was produced, inviting the readers to reach the destination in Grand Theft Auto V through maps and indications.

The game environment thus becomes the space for possible 'art tours', getting insights into the artworks made in GTA V. This form of game tourism allows the player to see the behind the scenes, and experience the making of the works in its place of origin. While the complex algorithms of GTA V produce unexpected interactions and scenes, Los Santos is also stuck in the same time forever. Gas stations, shops, palm trees remain in the same state and location forever, allowing the tourist to witness the exact scene that was first encountered by the artists.

Grand Theft Auto V Art Education

This project is also an attempt to introduce a video game as a space for artistic intervention, and an invitation to use its mechanics, its code and its environment as a creative tool itself. The game can be played, documented and captured through a form of artistic play, that differs from normative gameplay and does not focus on advancinf and winning but rather engages with the game object critically. Furthermore, the game software can also be manipulated, modified and used as an apparatus to create new images and interactions. The goal of this guide is to combine a curatorial approach that leads the viewer to discover the artworks made in GTA V with a hands on approach that teaches the player the tools for possible artistic interventions in this space.

Games are often seen as producing specific cultures and shaping identities through through forms of play that follow the intentions of the developer. Here we understand games as objects to be reclaimed and tools to be deconstructed and rebuilt, both conceptually and literally. Consequently, players are not just passive actors that push buttons in the sequence that they are taught by the machine and its softwares, but open up the black box of the game and become critical thinkers and makers that actively play with the game, or even against it.

The Photographer's Guide to Los Santos can be employed as a resource to accompany workshops for students and artists approaching computer games and interested in learning how to engage with it. Each thematic chapter features a tutorial section that introduces different techniques and strategies to capture images within Grand Theft Auto V, connected to the examples and locations of each section. The chapters are thought to be experienced in order, as the tutorials at times rely on knowledge that is built on top of previous lessons. Each tutorial is accompanied by content replication assignments, in which the readers is invited to use the skills learnd from each chapter to recreate a work presented in that section. Tutorials are intended for anyone who might be playing GTA V for the first time in their life and do not assume any previous experience, although some basic idea of programming is helpful when dealing with scripting and modding the game.

Chapter 2

Architecture Photography

//general intro on architecture in game spaces, the contruction of virtual cities, architecture photography and how it relates to the game environment, the player as a photographer documenting urban spaces...

The Continuous City, by Gareth Damian Martin

Gareth Damian Martin, *Outskirts*, from *The Continuous City*,

Gareth Damian Martin, *Pathways*, from *The Continuous City*,

artwork text

More about *The Continuous City*

Interview with Gareth Damian Martin

Getting there

- The intersection of Interstate 4 and Interstate 5 manifests the architecture of traffic of the megalopolis.

Readings

Heterotopias

Mark D Teo, The Urban Architecture of Los Angeles and Grand Theft Auto, 2015. https://www.academia.edu/18173221/The_Urban_Architecture_of_ Los_Angeles_and_Grand_Theft_Auto

Tutorial

Photographing the Game Screen

Analogue Game Photography

Screenshotting

Content Replication Assignment

Chapter 3

Social Documentary

//general intro on simulating society, the creation of NPCs, documentary and street photography traditions connected to politics of visibility and representation, and how they relate to the politics of simulation, how the player-photographer documents the creation of complex social spaces and reveals the process of simulating people and issues of class, gender, race in the game space...

Down and Out in Los Santos by Alan Butler

artwork text

More about *Down and Out in Los Santos*

Getting There

The homeless camp in Los Santos is under the Olympic Freeway in Strawberry.

Dignity Village is a tent city established by homeless people near Procopio Beach, east of Paleto Bay.

Fear and Loathing in GTA V by Morten Rockford Ravn

artwork text

More about *Fear and Loathing in GTA V*

Getting There

Readings

Tutorial

In-game Smartphone Camera

Snapmatic is the photo app on your simulated mobile phone in GTA V.

- Press `UP` on the d-pad to bring up your phone.

- Select the Snapmatic app - it's on the bottom left of the homescreen.

- You can shuffle through filters with `DOWN` on the d-pad or borders with `UP` on the d-pad.

- Move the camera with the `RIGHT STICK` and zoom in and out with the `LEFT STICK`.

- You're also able to concentrate focus and depth of field.

- To take selfie press the `R3 STICK` to turn the camera on yourself.

- `L3` will let you pull different facial expressions.

- `LEFT` on the d-pad will let you strike more of a pose (this changes depending on the character).

- Once you're happy, take the photo with `X` on the PS4 and `A` on the Xbox One and save it to the Gallery

Content Replication Assignment

Chapter 4

Re-enactment photography

//general introduction on the development of photorealism in games, the relationship between photography and CGI, the remediation of photographic images and the analog apparatus, the player as photographer situated in the tradition of conceptual photographers like Sherrie Levine and Sturtevant, the copy as a conceptual approach that create new meaning through a similar image but a different context...

26 Gasoline stations in GTA V by Lorna Ruth Galloway

artwork text

More about *26 Gasoline stations in GTA V*

Getting There

- Globe Oil Gas Station, Innocence Blvd & Alta St, South Los Santos
- LTD Gas Station, Davis Ave & Grove St, South Los Santos
- LTD Gas Station, Mirror Park Blvd & W Mirror Dr, Mirror Park
- Globe Oil Gas Station, Clinton Ave & Fenwell Pl, Vinewood Hills
- Xero Gas Station, Strawberry Ave & Capital Blvd, South Los Santos
- RON Gas Station, Davis Ave & Macdonald St, South Los Santos
- Xero Gas Station, Calais Ave & Innocence Blvd, Little Seoul
- LTD Gas Station, Lindsay Circus & Ginger St, Little Seoul
- RON Gas Station, N Rockford Dr & Perth St, Morningwood
- Xero Gas Station, Great Ocean Hwy, Pacific Bluffs

A Study on Perspective by Roc Herms

artwork text

More about *A Study on Perspective*

Getting There

Further references

Little Books of Los Santos by Luke Caspar Pearson

artwork text

More about *Little Books of Los Santos*

26 Gasoline stations in GTA V by M. Earl Williams

artwork text

More about *26 Gasoline stations in GTA V*

Readings

Tutorial

Scene Director Mode

Content Replication Assignment

Chapter 5

Nature Documentary

//general introduction about the creation of a synthetic forms of nature, ecological issues, creation of virtual sublime, flora and fauna that are usually props that become the focus of the player's explorations, "virtual world naturalism"...

San Andreas Streaming Deer Cam by Brent Watanabe

artwork text

More about *Deercam*

Getting There

Mount Chiliad is located in the Chiliad Mountain State Wilderness, and it is the tallest mountain in the game at 798m above sea level. The state park is home to lots of wildlife such as deer and mountain lions.

Virtual Botany Cyanotype by Alan Butler

//selectioon of flora from GTA V

artwork text

More about *Virtual Botany Cyanotype*

Getting There

Readings

Tutorial

Scripting Introduction

Preparation and Setup

- Install Windows 11

- Download and install Steam (with a copy of GTA V or buy the game if you do not have it. GTA V is 100+ GB so it will take a few hours depending on your internet connections)

- Download Script Hook V, go to the bin folder and copy `dinput8.dll` and `ScriptHookV.dll` files into your GTA V directory `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`

- Download Script Hook V dot net, copy the `ScriptHookVDotNet.asi` file, `ScriptHookVDotNet2.dll` and `ScriptHookVDotNet3.dll` files into your GTA V directory `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`

- Create a new folder in GTA V directory and call it "scripts".

- Download and install Visual Studio Community (free version of VS). Open Visual Studio and check the .NET desktop development package and install it

- Run GTA V and test if Script Hook V is working by pressing `F4`. This should toggle the console view.

Creating a Mod File

- Open Visual Studio

- Select File > New > Project

- Select Visual C# and Class Library (.NET Framework)

- Give a custom file name (e.g. moddingTutorial)

- Rename public class Class1 as "moddingTutorial" in the right panel Solution Explorer

- In the same panel go to References and click add References... > Browse > browse to Downloads

- Select ScriptHookedVDotNet > `ScriptHookVDotNet2.dll` and `ScriptHookVDotNet3.dll` and add them

- Also add `System.Windows.forms`

- Also add `System.Drawing`

- In your code file add the following lines on top:

```
using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
```

- Modify class moddingTutorial to the following:

```
namespace moddingTutorial
{
  public class moddingTutorial : Script
  {
      public moddingTutorial()
      {
          this.Tick += onTick;
      this.KeyUp += onKeyUp;
      this.KeyDown += onKeyDown;
      }

      private void onTick(object sender, EventArgs e)
      {
      }

      private void onKeyUp(object sender, KeyEventArgs e)
      {
      }

      private void onKeyDown(object sender, KeyEventArgs e)
      {
        if (e.KeyCode == Keys.H)
        {
            Game.Player.ChangeModel(PedHash.Cat);
          }
      }
```

```
   }
}
```

- Save file

- Go to Documents > Visual Studio > Project > moddingTutorial > moddingTutorial > `moddingTutorial.cs`

- Copy the .cs file in the GTA V directory inside the scripts folder

- Open GTA V, run the game in Story Mode (mods are only allowed in single player mode, not in GTA Online) and press 'H' to see if the game turns your avatar into a cat

- Note: every time you make changes to your .cs file in the scripts folder you can hit `F4` to open the console, type `Reload()` in the console for the program to reload the script and test again the changes.

onTick, onKeyUp and onKeyDown

The main events of Script Hook V Dot Net are onTick, onKeyUp and onKeyDown. Script Hook V Dot Net will invoke your functions whenever an event is called.

The code within the onTick brackets is executed every interval milliseconds (which is by default 0), meaning that the event will be executed at every frame, for as long as the game is running.

```
private void onTick(object sender, EventArgs e)
{
        //code here will be executed every frame (or per usef defined interval)
}
```

If your function is written inside onKeyDown (withiin the curly brackets following onKeyUp(object sender, KeyEventArgs e){}), your code will be executed every time a key is pressed. If your function is written inside onKeyUp, your code will be executed every time a key is released.

```
private void onKeyUp(object sender, KeyEventArgs e)
{
      //code here will be executed whenever a key is released
}


private void onKeyDown(object sender, KeyEventArgs e)
{
      //code here will be executed whenever a key is pressed
}
```

We can specify which code is executed based on what keys are pressed/released

```csharp
private void onKeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.H)
    {
        //code here will be executed whenever the key 'H' is pressed
    }
}
```

Change Player Model

The player character is controlled as Game.Player. Game.Player can perform different functions, including changing the avatar model, and performing tasks.

Change the 3D model of your character by using the `ChangeModel` function. The function needs a model ID, in order to load the model file of our game character. You can browse through this list of models to find the one you want to try: https://wiki.gtanet.work/index.php/Peds

These models are all PedHashes, basically ID numbers within the PedHash group. Copy the name of the model below the image and add it to PedHash. For example if you choose the model Cat, you'll need to write `PedHash.Cat`.

To change the model of your player character into a cat you can write the following function:

```csharp
Game.Player.ChangeModel(PedHash.Cat);
```

add it in your .cs file in the onKeyDown event, triggered by the pressing of the 'h' key:

Example code

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

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
```

```
namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed
        {
        }

        private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is
        {
        }

        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here
        {
            //when pressing 'H'
            if(e.KeyCode == Keys.H)
            {
                //change player char into a different model
                Game.Player.ChangeModel(PedHash.Cat);
            }
        }
    }
}
```

Tasks

Our character can be controlled by our script, and given actions that override manual control of the player. These actions are called *Tasks* and in order to assign tasks to our characters we have to define our `Game.Player` as `Game.Player.Character`. The `Game.Player.Character` code gets the specific model the player is controlling.

Now we can give tasks to the character by adding the `Task` function: `Game.Player.Character.Task`.

Finally we can specify what task to give the character by choosing a task from TaskInvoker list of possible actions.

Jump:

```
Game.Player.Character.Task.Jump();
```

Wander around:

```
Game.Player.Character.Task.WanderAround();
```

Hands up for 3000 milliseconds:

```
Game.Player.Character.Task.HandsUp(3000);
```

Task Sequences

You can create sequence of multiple tasks by using `TaskSequence` and the `PerformSequence` function. Create a new `TaskSequence` with a custom name, add tasks to it with `AddTask`, close the sequence with `Close` and then call `Task.PerformSequence` to perform the sequence.

```
TaskSequence mySeq = new TaskSequence();
mySeq.AddTask.Jump();
mySeq.AddTask.HandsUp(3000);
mySeq.Close();

Game.Player.Character.Task.PerformSequence(mySeq);
```

Random

We can add randomness by using a randomly generated number, which makes things outside of thepredefined programme controlled by us and introduces more autonomous behaviours. We use the `Random`function to create a randomly generated number between our minimum and maximum parameter (if only one parameter is inserted, the minimum is 0).

```
Random rnd = new Random();
int month = rnd.Next(1, 13); // creates a number between 1 and 12
int dice = rnd.Next(1, 7); // creates a number between 1 and 6
int card = rnd.Next(52); // creates a number between 0 and 51
```

Let's create a number to generate a random duration between 1 and 6 seconds, for the `HandsUp` task.

```
Random rnd = new Random();
int waitingTime = rnd.Next(1, 7);
Game.Player.Character.Task.HandsUp(waitingTime * 1000);
```

Subtitles and Notifications

Generate subtitles with a custom text string and duration (in milliseconds):

```
UI.ShowSubtitle("Hello World", 3000);
```

Generate a notification with a custom text string:

```
UI.Notify("Hello World");
```

Content Replication Assignment

Deercam reenactment

Write a mod script to change your game character into a deer by pressing a key, and make it autonomously wander around Los Santos by pressing another key.

Chapter 6

Surrealist Photography

//general introduction about avant garde traditions of distancing from reality and exploring the possibilities of CGI decoupled from realism adn life-like simulation, the game as an engine that can be used to create oniric scenes, which in turn reveal the untapped possibilities hidden within the game code, the player as a modder which can generate worlds within the world...

Alexey Andrienko aka HAPP v2

artwork text

More about Happ v2

Getting There

Chumash Beach

Readings

Tutorial

Scripting Characters

NPCs

NPCs are non playable characters and in GTA V scripting they are called `Peds`. Peds are an entity like Props or Vehicles and can be created, assigned different

model textures, equipped with weapons and controlled through different tasks.

Spawn a new NPC

A GTA V Ped can be created by the `World.CreatePed` function. This takes two parameters: an ID to assign the 3D model and textures, and the location where the Ped is created.

The model IDs are the same we used in the previous tutorial, when we changed our character's appearance to a cat. A list of all available models can be found here. `PedHash.Cat`, `PedHash.Deer`, `PedHash.AviSchwartzman`are all possible IDs we can assign to the NPC we want to create. We can create a new model variable, which we will name 'myPedModel' and assign it a model ID:

```
Models myPedModel = "PedHash.AviSchwartzman";
```

The location where the NPC is created through a vector3 data type, which represents a vector in 3D space. This basically means a point that contains X, Y and Z coordinates. We can give absolute coordinates, making the Ped appear at a specific location in the game, but we can also use a location relative to our position in the game. In order not to risk making a Ped appear somewhere completely outside of our view – on some mountain or in the sea – let's look at a vector3 that points to a position in front of the player.

We want to establish the player with`Game.Player.Character`, followed by a function that retireve the player position within the game world. That's called by using `GetOffsetInWorldCoords`, which takes a vector3. The values of the X, Y and Z of the vector 3 offset the location based on the origin point represented by the player. Therefore, we can move the place where we want the Ped to appear by adding values to the X axis (left or right of player), Y axis (ahead or behind the player), and Z axis (above or below the player). To make a Ped appear in front of the player we can create a vector3 data type with 0 for X, 5 for Y and 0 for Z: `new Vector3(0, 5, 0)`. Let's make a vector3 variable, which we will name 'myPedSpawnPosition', assign it the values above for X, Y and Z coordinates from the player position.

```
Vector3 myPedSpawnPosition = Game.Player.Character.GetOffsetInWorldCoords(new Vector3(
```

Now we can use the model and the position variables to spawn the NPC in front of the player. We'll create a Ped named 'myPed1' and use the `World.CreatePed` function with the two variables as parameters:

```
var myPed1 = World.CreatePed(myPedModel, myPedSpawnPosition);
```

Example code

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

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
            this.KeyDown += onKeyDown;
        }

        private void onTick(object sender, EventArgs e) //this function gets executed continuousl
        {
        }

        private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is executed o
        {
        }

        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here is execute
        {
            //when pressing 'K'
            if(e.KeyCode == Keys.K)
            {
                //select a model and store it in a variable
                Models myPedModel = "PedHash.AviSchwartzman";
                    //create a position relative to the player
                    Vector3 myPedSpawnPosition = Game.Player.Character.GetOffsetInWorldCoords(new
                    //create a Ped with the chosen model, spawning at the chosen position
                    var myPed1 = World.CreatePed(myPedModel, myPedSpawnPosition);
            }
        }
```

```
    }
}
```

Control Multiple NPCs

You can create multiple NPCs and give them custom names. Let's create a human NPC and a cat NPC and call them Jim and MannyTheCat respectively:

```
var Jim = World.CreatePed(PedHash.AviSchwartzman, Game.Player.Character.GetOffsetInWorl
var MannyTheCat = World.CreatePed(PedHash.Cat, Game.Player.Character.GetOffsetInWorldCo
```

Try to kill one of the `Ped` NPCs you created by using the `Kill()`.

```
Jim.Kill();
```

Note that when you kill your `Ped` 'Jim', it falls on the floor and it won't actually respond to any call or task you will give it, but it's not removed from the game. To remove a specific `Ped` you not use the `Delete`function, which will remove that instance (and will make the NPC disappear).

```
Jim.Delete();
```

To handle groups of NPCs we can use the `List` class. A `List` is a collection of objects, and a `List`of `Peds` allows us to store our NPCs. We can use an index to retrieve and control specific `Peds` in the group. ou can see the reference for more detailed information.

Create a `List` of `Peds` named myPeds as a global variable in the public class `public class moddingTutorial : Script`.

```
List<Ped> myPeds = new List<Ped>();
```

In the onKeyDown function `private void onKeyDown(object sender, KeyEventArgs e)` create 5 new Peds with a For Loop

```
for (int i = 0; i < 5; i++)
{
    //spawn a new Ped called newPed
    var newPed = World.CreatePed(PedHash.AviSchwartzman, Game.Player.Character.GetOffse
    //add the new Ped to my list of Peds myPeds
    myPeds.Add(newPed);
}
```

Now all the 5 `Peds` are part of the myPeds[] `List`. You can control each Ped individually by calling their individual number ID in the group. The first spawn `Ped` is myPed[0], the last one is myPeds[4].

Kill the 3rd spawned NPC:

```
myPeds[2].Kill();
```

Nearby NPCs

Script Hook V DOt Net provides a function `GetNearbyPeds`which groups all the `Peds` within a nearby radius from a character.

Create a new group that adds `Peds` which are closer than 20 meters from the player:

```
Ped[] NearbyPeds = World.GetNearbyPeds(Game.Player.Character, 20f);
```

Use a Foreach Loop to get every `Ped` in the group and kill them:

```
foreach (Ped p in NearbyPeds)
{
    p.Kill();
}
```

`GetNearbyPeds` does not sort out individual `Peds`in the group based on distance, so we have to do a bit of manual filtering to get the nearest NPC within the chosen radius from the player character.

Define the global variables in the public class `public class moddingTutorial : Script`:

```
float lastDistance;
Ped Squad1Leader = null;
Ped oldSquad1Leader = null;
```

Get and parse the nearby NPCs in the OnTick function `private void onTick(object sender, EventArgs e)`:

```
//set radius
float maxDistance = 25f;
//get nearest peds
Ped[] pedsGroup = World.GetNearbyPeds(Game.Player.Character, maxDistance);
```

```
float lastDistance = maxDistance;
foreach (Ped ped in pedsGroup)
{
    float distance = ped.Position.DistanceTo(Game.Player.Character.Position);
    if (distance < lastDistance)
    {
        Squad1Leader = ped;
        lastDistance = distance;
    }
}

    if (Squad1Leader != null && oldSquad1Leader != Squad1Leader)
    {
        Squad1Leader.Kill();
    }
oldSquad1Leader = Squad1Leader;
```

Give Tasks to NPCs

A Ped can be given a task using the `Task` function, just like we did in the
previous tutorial for the player character.

```
myPed1.Task.WanderAround();
```

Some tasks involve interacting with other characters (Peds or Game.Player.Character)
or take different parameters like positions (vector3), duration (in milliseconds),
and other data types. We can give our NPC the task to fight against the player
by using the `FightAgainst` function, which requires a Ped parameter – which
in the case of the player is expressed as `Game.Player.Character`.

```
myPed1.Task.FightAgainst(Game.Player.Character); //give npc task to fight against play
```

Try to replace the task to "fight against" with "flee from (player)" , "hands up",
"jump"... or some of the other available tasks.

See the TaskInvoker list for possible tasks, or click on the list of available tasks
below.

List of Available Tasks

```
void AchieveHeading (float heading, int timeout=0)
```

```
void AimAt (Entity target, int duration)
```

```csharp
void AimAt (Vector3 target, int duration)

void Arrest (Ped ped)

void ChatTo (Ped ped)

void Jump ()

void Climb ()

void ClimbLadder ()

void Cower (int duration)

void ChaseWithGroundVehicle (Ped target)

void ChaseWithHelicopter (Ped target, Vector3 offset)

void ChaseWithPlane (Ped target, Vector3 offset)

void CruiseWithVehicle (Vehicle vehicle, float speed, DrivingStyle style=DrivingStyle.Normal)

void DriveTo (Vehicle vehicle, Vector3 target, float radius, float speed, DrivingStyle style=Driv

void EnterAnyVehicle (VehicleSeat seat=VehicleSeat.Any, int timeout=-1, float speed=1f, EnterVehi

void EnterVehicle (Vehicle vehicle, VehicleSeat seat=VehicleSeat.Any, int timeout=-1, float speed

void FightAgainst (Ped target)

void FightAgainst (Ped target, int duration)

void FightAgainstHatedTargets (float radius)

void FightAgainstHatedTargets (float radius, int duration)

void FleeFrom (Ped ped, int duration=-1)

void FleeFrom (Vector3 position, int duration=-1)

void FollowPointRoute (params Vector3[] points)

void FollowPointRoute (float movementSpeed, params Vector3[] points)

void FollowToOffsetFromEntity (Entity target, Vector3 offset, float movementSpeed, int timeout=-1
```

```
void GoTo (Entity target, Vector3 offset=default(Vector3), int timeout=-1)

void GoTo (Vector3 position, int timeout=-1)

void GoStraightTo (Vector3 position, int timeout=-1, float targetHeading=0f, float dist

void GuardCurrentPosition ()

void HandsUp (int duration)

void LandPlane (Vector3 startPosition, Vector3 touchdownPosition, Vehicle plane=null)

void LeaveVehicle (LeaveVehicleFlags flags=LeaveVehicleFlags.None)

void LeaveVehicle (Vehicle vehicle, bool closeDoor)

void LeaveVehicle (Vehicle vehicle, LeaveVehicleFlags flags)

void LookAt (Entity target, int duration=-1)

void LookAt (Vector3 position, int duration=-1)

void ParachuteTo (Vector3 position)

void ParkVehicle (Vehicle vehicle, Vector3 position, float heading, float radius=20.0f

void PerformSequence (TaskSequence sequence)

void PlayAnimation (string animDict, string animName)

void PlayAnimation (string animDict, string animName, float speed, int duration, float

void PlayAnimation (string animDict, string animName, float blendInSpeed, int duration

void PlayAnimation (string animDict, string animName, float blendInSpeed, float blendOu

void RappelFromHelicopter ()

void ReactAndFlee (Ped ped)

void ReloadWeapon ()

void RunTo (Vector3 position, bool ignorePaths=false, int timeout=-1)

void ShootAt (Ped target, int duration=-1, FiringPattern pattern=FiringPattern.Default)
```

```csharp
void ShootAt (Vector3 position, int duration=-1, FiringPattern pattern=FiringPattern.Default)

void ShuffleToNextVehicleSeat (Vehicle vehicle=null)

void Skydive ()

void SlideTo (Vector3 position, float heading)

void StandStill (int duration)

void StartScenario (string name, float heading)

void StartScenario (string name, Vector3 position, float heading)

void SwapWeapon ()

void TurnTo (Entity target, int duration=-1)

void TurnTo (Vector3 position, int duration=-1)

void UseParachute ()

void UseMobilePhone ()

void UseMobilePhone (int duration)

void PutAwayParachute ()

void PutAwayMobilePhone ()

void VehicleChase (Ped target)

void VehicleShootAtPed (Ped target)

void Wait (int duration)

void WanderAround ()

void WanderAround (Vector3 position, float radius)

void WarpIntoVehicle (Vehicle vehicle, VehicleSeat seat)

void WarpOutOfVehicle (Vehicle vehicle)

void ClearAll ()
```

```
void ClearAllImmediately ()

void ClearLookAt ()

void ClearSecondary ()

void ClearAnimation (string animSet, string animName)
```

Animations

//TO DO

Teleporting

We can change the location of the player character or of any Ped or Vehicle entity by using the native function `SET_ENTITY_COORDS`. This function needs an entity and X, Y and Z coordinate to teleport to. We need to know the exact coordinates of the locations we want to teleport to, but thankfully the modding community forums provide lists with all available coordinates we can teleport to. Let's take the XYZ coordinates of the top of Mount Chiliad (the highest point in the game) to teleport our player character to.

```
LOCATION: Top of the Mt Chilad
COORDINATES: X:450.718 Y:5566.614 Z:806.183
```

To create a teleport function we will use a native function. Script Hook V Dot Net is a wrapper for the C++ ScriptHook, calling the functions in Scripthook to do things in the game. However, there are some functions that are not in Script Hook V Dot Net and in order to use these, we have to use the native calling from Script Hook.

Native functions are called with `Function.Call` followed by their corresponding hash name and parameters. They use this structure:

```
Function.Call(Hash.HASH_NAME, input_params);
```

The native function for teleporting expects the hash `SET_ENTITY_COORDS`, the `ped` entity to teleport, and the X, Y and Z coordinates to teleport the character to. `Function.Call(Hash.SET_ENTITY_COORDS, Ped ped, X, Y, Z, 0, 0, 1);`

The function to teleport the player character to the top of Moutn Chiliad is:

```
//Teleport to the top of Mount Chiliad
Function.Call(Hash.SET_ENTITY_COORDS, Game.Player.Character, 450.718f, 5566.614f, 806.183f, 0, 0,
```

See this list of locations to find their respective coordinates or click on the list below

List of Locations with Coordinates

```
INDOOR LOCATIONS

Strip Club DJ Booth X:126.135 Y:-1278.583 Z:29.270

Blaine County Savings Bank X:-109.299 Y:6464.035 Z:31.627

Police Station X:436.491 Y: -982.172 Z:30.699

Humane Labs Entrance X:3619.749 Y:2742.740 Z:28.690

Burnt FIB Building X:160.868 Y:-745.831 Z:250.063

10 Car Garage Back Room X:223.193 Y:-967.322 Z:99.000

Humane Labs Tunnel X:3525.495 Y:3705.301 Z:20.992

Ammunation Office X:12.494 Y:-1110.130 Z: 29.797

Ammunation Gun Range X: 22.153 Y:-1072.854 Z:29.797

Trevor's Meth Lab X:1391.773 Y:3608.716 Z:38.942

Pacific Standard Bank Vault X:255.851 Y: 217.030 Z:101.683

Lester's House X:1273.898 Y:-1719.304 Z:54.771

Floyd's Apartment X:-1150.703 Y:-1520.713 Z:10.633

FIB Top Floor X:135.733 Y:-749.216 Z:258.152

IAA Office X:117.220 Y:-620.938 Z:206.047

Pacific Standard Bank X:235.046 Y:216.434 Z:106.287

Fort Zancudo ATC entrance X:-2344.373 Y:3267.498 Z:32.811

Fort Zancudo ATC top floor X:-2358.132 Y:3249.754 Z:101.451
```

```
Torture Room X: 147.170 Y:-2201.804 Z:4.688
```

```
OUTDOOR LOCATIONS
```

```
Main LS Customs X:-365.425 Y:-131.809 Z:37.873
```

```
Very High Up X:-129.964 Y:8130.873 Z:6705.307
```

```
IAA Roof X:134.085 Y:-637.859 Z:262.851
```

```
FIB Roof X:150.126 Y:-754.591 Z:262.865
```

```
Maze Bank Roof X:-75.015 Y:-818.215 Z:326.176
```

```
Top of the Mt Chilad X:450.718 Y:5566.614 Z:806.183
```

```
Most Northerly Point X:24.775 Y:7644.102 Z:19.055
```

```
Vinewood Bowl Stage X:686.245 Y:577.950 Z:130.461
```

```
Sisyphus Theater Stage X:205.316 Y:1167.378 Z:227.005
```

```
Galileo Observatory Roof X:-438.804 Y:1076.097 Z:352.411
```

```
Kortz Center X:-2243.810 Y:264.048 Z:174.615
```

```
Chumash Historic Family Pier X:-3426.683 Y:967.738 Z:8.347
```

```
Paleto Bay Pier X:-275.522 Y:6635.835 Z:7.425
```

```
God's thumb X:-1006.402 Y:6272.383 Z:1.503
```

```
Calafia Train Bridge X:-517.869 Y:4425.284 Z:89.795
```

```
Altruist Cult Camp X:-1170.841 Y:4926.646 Z:224.295
```

```
Maze Bank Arena Roof X:-324.300 Y:-1968.545 Z:67.002
```

```
Marlowe Vineyards X:-1868.971 Y:2095.674 Z:139.115
```

```
Hippy Camp X:2476.712 Y:3789.645 Z:41.226
```

```
Devin Weston's House X:-2639.872 Y:1866.812 Z:160.135
```

Abandon Mine X:-595.342 Y: 2086.008 Z:131.412

Weed Farm X:2208.777 Y:5578.235 Z:53.735

Stab City X: 126.975 Y:3714.419 Z:46.827

Airplane Graveyard Airplane Tail X:2395.096 Y:3049.616 Z:60.053

Satellite Dish Antenna X:2034.988 Y:2953.105 Z:74.602

Satellite Dishes X: 2062.123 Y:2942.055 Z:47.431

Windmill Top X:2026.677 Y:1842.684 Z:133.313

Sandy Shores Building Site Crane X:1051.209 Y:2280.452 Z:89.727

Rebel Radio X:736.153 Y:2583.143 Z:79.634

Quarry X:2954.196 Y:2783.410 Z:41.004

Palmer-Taylor Power Station Chimney X: 2732.931 Y: 1577.540 Z:83.671

Merryweather Dock X: 486.417 Y:-3339.692 Z:6.070

Cargo Ship X:899.678 Y:-2882.191 Z:19.013

Del Perro Pier X:-1850.127 Y:-1231.751 Z:13.017

Play Boy Mansion X:-1475.234 Y:167.088Z:55.841

Jolene Cranley-Evans Ghost X:3059.620 Y:5564.246 Z:197.091

NOOSE Headquarters X:2535.243 Y:-383.799 Z:92.993

Snowman X: 971.245 Y:-1620.993 Z:30.111

Oriental Theater X:293.089 Y:180.466 Z:104.301

Beach Skatepark X:-1374.881 Y:-1398.835 Z:6.141

Underpass Skatepark X:718.341 Y:-1218.714 Z: 26.014

Casino X:925.329 Y:46.152 Z:80.908

University of San Andreas X:-1696.866 Y:142.747 Z:64.372

La Puerta Freeway Bridge X: -543.932 Y:-2225.543 Z:122.366

Land Act Dam X: 1660.369 Y:-12.013 Z:170.020

Mount Gordo X: 2877.633 Y:5911.078 Z:369.624

Little Seoul X:-889.655 Y:-853.499 Z:20.566

Epsilon Building X:-695.025 Y:82.955 Z:55.855 Z:55.855

The Richman Hotel X:-1330.911 Y:340.871 Z:64.078

Vinewood sign X:711.362 Y:1198.134 Z:348.526

Los Santos Golf Club X:-1336.715 Y:59.051 Z:55.246

Chicken X:-31.010 Y:6316.830 Z:40.083

Little Portola X:-635.463 Y:-242.402 Z:38.175

Pacific Bluffs Country Club X:-3022.222 Y:39.968 Z:13.611

Vinewood Cemetery X:-1659993 Y:-128.399 Z:59.954

Paleto Forest Sawmill Chimney X:-549.467 Y:5308.221 Z:114.146

Mirror Park X:1070.206 Y:-711.958 Z:58.483

Rocket X:1608.698 Y:6438.096 Z:37.637

El Gordo Lighthouse X:3430.155 Y:5174.196 Z:41.280

Content Replication Assignment

Teleport the player to a beach, spawn ten whales on the shore and generate an NPC wandering aroud them and take a screenshot in the style of HAPP V2.

Chapter 7

Hyperrealist Photography

//general introduction on the relationship between the game and the physical world, the way virtual spaces influence and shape society (training self driving cars in GTA V, CGI shaping architecture of buildings and object...), the blurrying of the lines between virtual and physical...

8k by Aram Bartholl

Aram Bartholl, *8k*, installation view

Aram Bartholl, *8k*, installation view

More about 8k

Getting there

- Land Act Dam, Tataviam Mountains

Readings

Tutorial

Setting Camera Views

GTA V has 4 default camera views, which can be switched by pressing the `V` key on PC. To set a specific camera view we can use the native function `SET_FOLLOW_PED_CAM_VIEW_MOD`, followed by number 0, 1, 2 or 4 to establish the desired point of view:

- 0 - Third Person View - Close
- 1 - Third Person View - Mid
- 2 - Third Person View - Far
- 4 - First Person View

Switch to first person view:

```
Function.Call(Hash.SET_FOLLOW_PED_CAM_VIEW_MODE, 4);
```

Controlling the Game Camera

Create Multiple Cameras

Attach a Camera to an Entity

Example code

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

using GTA;
using GTA.Math;
using System.Windows.Forms;
using System.Drawing;
using GTA.Native;
using System.IO;


namespace moddingTutorial
{
    public class moddingTutorial : Script
    {
        Vector3 myCamPos;
        int CamSelect = 0;
        Ped newPed = null;
        Camera myCam;

        public moddingTutorial()
        {
            this.Tick += onTick;
            this.KeyUp += onKeyUp;
```

```csharp
        this.KeyDown += onKeyDown;
    }

    private void onTick(object sender, EventArgs e) //this function gets executed continuousl
    {
        //exits from the loop if the game is loading
        if (Game.IsLoading) return;

        //update the cameras if the ped is spawn
        if (newPed != null)
        {
            //create the cameras if none have been created yet.
            if (myCam == null)
            {
                UI.ShowSubtitle("Set new camera");
                myCam = World.CreateCamera(Vector3.Zero, newPed.Rotation, 50f);
                // Set the camera position (relative pos)
                myCamPos = new Vector3(0, 0, 1f);
            }
            //attach the cameras
            myCam.AttachTo(newPed, myCamPos);
            //sync rotation
            myCam.Rotation = newPed.Rotation;
        }
    }


    private void onKeyUp(object sender, KeyEventArgs e)//everything inside here is executed o
    {
        //press control+K to switch between gameplay default camera and the NPC camera
        if (e.KeyCode == Keys.K && e.Modifiers == Keys.Shift && newPed != null)
        {
            CamSelect = (CamSelect + 1) % 2;
            switch (CamSelect)
            {
                case 0: World.RenderingCamera = null;
                    UI.ShowSubtitle("Showing Gameplay Cam View");
                    break;
                case 1: World.RenderingCamera = myCam;
                    UI.ShowSubtitle("Showing NPC Cam View");
                    break;
            }

        }
    }
```

```csharp
        private void onKeyDown(object sender, KeyEventArgs e) //everything inside here
        {
            if(e.KeyCode == Keys.G)
            {
                //spawn new Ped
                newPed = World.CreatePed(PedHash.Cat, Game.Player.Character.GetOffsetI
            }

            if (e.KeyCode == Keys.H)
            {
                //follow player (persistent)
                Function.Call(Hash.TASK_FOLLOW_TO_OFFSET_OF_ENTITY, newPed.Handle, Game
                //look at player
                newPed.Task.LookAt(Game.Player.Character);
            }
            if (e.KeyCode == Keys.J)
            {
                //stop NPC
                newPed.Task.ClearAll();
            }
            if (e.KeyCode == Keys.L)
            {
                //delete ped
                newPed.Delete();
            }
        }
    }
}
```

Scripting Cinematic Fade Out/ In

Script Hook has native functions to create a fead to/from black. The function
hash are `DO_SCREEN_FADE_OUT` and `DO_SCREEN_FADE_IN` and they are followed
by the number of milliseconds to go from full black to showing the scene and
viceversa.

Let's create a fade to black over 3 seconds when we press the letter key 'O':

```csharp
if (e.KeyCode == Keys.O)
{
    Function.Call(Hash.DO_SCREEN_FADE_OUT, 3000);
}
```

And a fade over 3 seconds when we press the letter key 'I':

```
if (e.KeyCode == Keys.I)
{
    Function.Call(Hash.DO_SCREEN_FADE_IN, 3000);
}
```

We could also create an automated check in our onTick loop, which keeps seeing if the screen has been faded to black. We can use the native function `IS_SCREEN_FADED_OUT` which is a boolean data type. This means it will return either true or false. If it returns true, it means the screen has been faded out.

Let's add an If statement in our onTick loop to chek if the screen has been faded to black, and if so we call a fade in over 3 second:

```
if (Function.Call<bool>(Hash.IS_SCREEN_FADED_OUT))
{
    Wait(500);
    GTA.Native.Function.Call(Hash.DO_SCREEN_FADE_IN, 3000);
}
```

Now if you try to hit the 'O' key, the screen will fade out, and then it will automatically fade in again.

Natural Vision Evolved Mod

Natural Vision Evolved is a graphic mod develped by Jamal Rashid, aka Razed. This mod enhances GTA V's lighting, weather effects, ambient colours, world textures, building models, pushing the photo-realism and cinematic looks. While the mod contains settings for different hardware settings, it's recommended to have a relatively powerful PC with a good graphic card. Here you can find minimum and recommended requirements for Natural Vision Evolved mod.

Setup:

- Go to razedmods.com/gta-v and download Natural Vision Evolved (6.2 Gb).

- Go to openiv.com/ and download Open IV, Open 'ovisetup' and install Open IV on your computer.

- Open the Open IV app and select GTA V Windows. Choose Grand Theft Auto V folder `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`

- Once Open IV is open, go to your file window select the `Tools` menu on top of the window, and select `ASI Manager`. In `ASI Manager` install all options: ASI Loader, OpenIV.ASI and openCamera.

- Select `Tools` again and click `Options`. Click on the `"mods" folder` tab and select `Allow edit mode only for archive inside "mods" folder`. Click `Close`.

- Back in Open IV, select `Edit mode` at the top right of the window. Select `OK` on the pop up window.

- Now you can add your mod to the mods folder.

- Open the NVE mod folder and copy the subfolder containing the settings you want (low/medium/ultra graphics) to your GTA V directory folder `C:\Program Files (x86)\Steam\steamapps\common\Grand Theft Auto V`.

- Go back to your downloads and inside the NVE folder select the installation files and open number 1 and 2 with Open IV in the correct order. Always choose to select "mod" folder and select install.

- Go back to your downloads and inside the NVE folder you can choose some optional addons. Install/Uninstall them with Open IV as above Always choose to select "mod" folder and select install.

- Open GTA V, press `ESC` to bring up the menu and go to `SETTINGS`. Adjust the graphics quality, making sure `Shader Quality`, `Particle Quality` and `Post FX`are set to `<Very High>`. Restart the game to make change in effect.

Content Replication Assignment