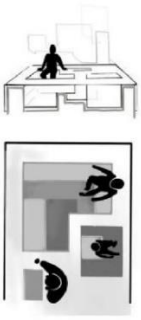
CHRONOSCAPE

Jiangyun Pan & Yiren Wang

Abstract contours form single rods that suspend/insert semi-transparent fabric within each hollow space. High degrees of freedom in form.
Inverted world: It can be the reflection of any space, achieving the transformation between physical and digital spaces.



From a top-down perspective, it is a space that extends downward, with the outline of the missing space serving as a miniature representation of the city/ space.
People explore and fill the space, leaving traces. The framework with the bottom hollowed out allows for better realization.

Using nets to gather sand/other objects in a limited space symbolizes the condensation of temporal space. The gathered sand carries weight, representing the existence of people and objects in the world. The overall gathering can be achieved through bundling, exerting a certain control over the form of the object. In this way, the "imprints" of people will also change the form of the object without altering its essence.

The camera captures human behavior in the physical space from the top and projects it into the digital world. The digital world corresponds to the physical world, representing a transformation of spatial relationships in terms of up, down, front, and back.

The exploration of the unknown in the physical space will be concretely reflected in the digital world - imprints.

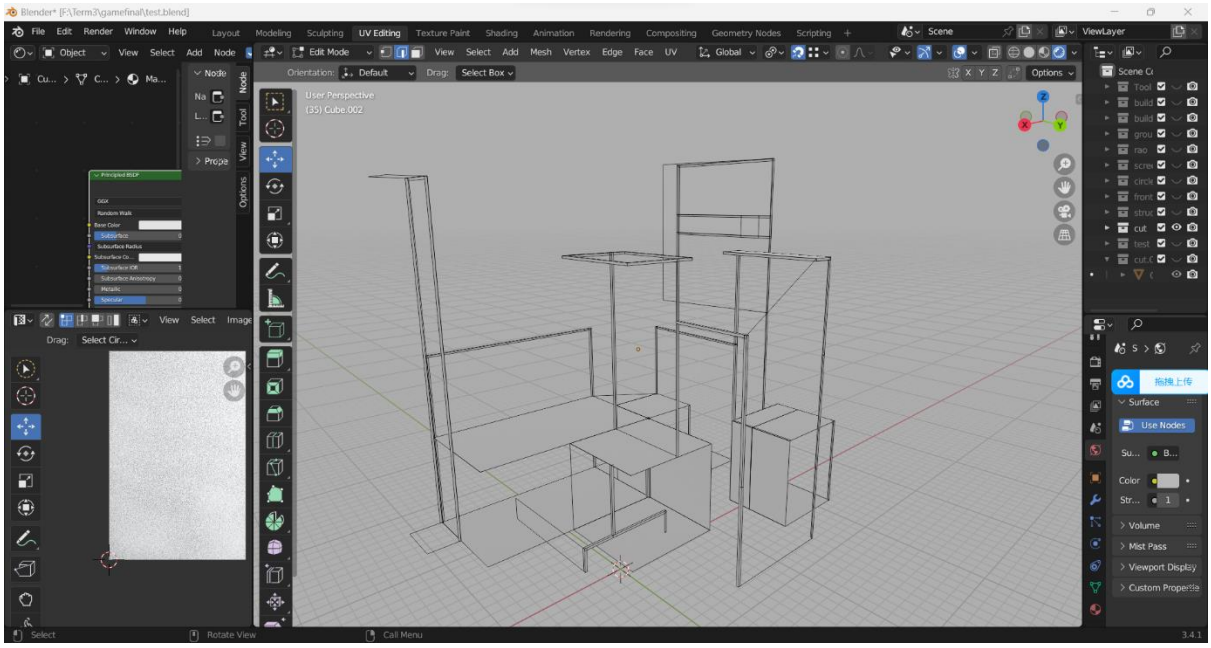
Digital World Concept:
Color Unification. A pure white memory world, ethereal and blurred.

In a world where most individuals leave minimal traces during their lifetimes, their absence becomes an irrevocable reality. However, as time inevitably slips away, people yearn to preserve their fleeting impact on the world around them. Yet, the footprints they leave behind fade into obscurity with the passage of time. In this intangible realm, how can one's existence extend into the digital domain? Hence, we embark on the creation of an everlasting memorial world—a frozen testament to existence and the passage of time.

I created a digital world that the imagery has a certain level of surrealism, reflecting the erosion of time. I chose objects such as buildings, screens, stairs, gates, curtains, and benches to compose the scene, all of which are closely related to people's lives. Everything in the scene can be floating, distorted, and seemingly unrelated objects are interconnected, forming a cycle. This world is like an imbalanced yet equal amusement park for humanity.

Our objective is to design an immersive playground that magnifies and commemorates the impact individuals have made. We have constructed a physical setting that mirrors a virtual world in terms of its structure and form. By deploying neural networks, we are able to track the positions of people in the real world and project their movements onto the virtual realm. This experimental mixed-reality game blends the realms of neural networks and reality. The aim for our players is to leave behind a multitude of captivating and significant traces within this interconnected world.

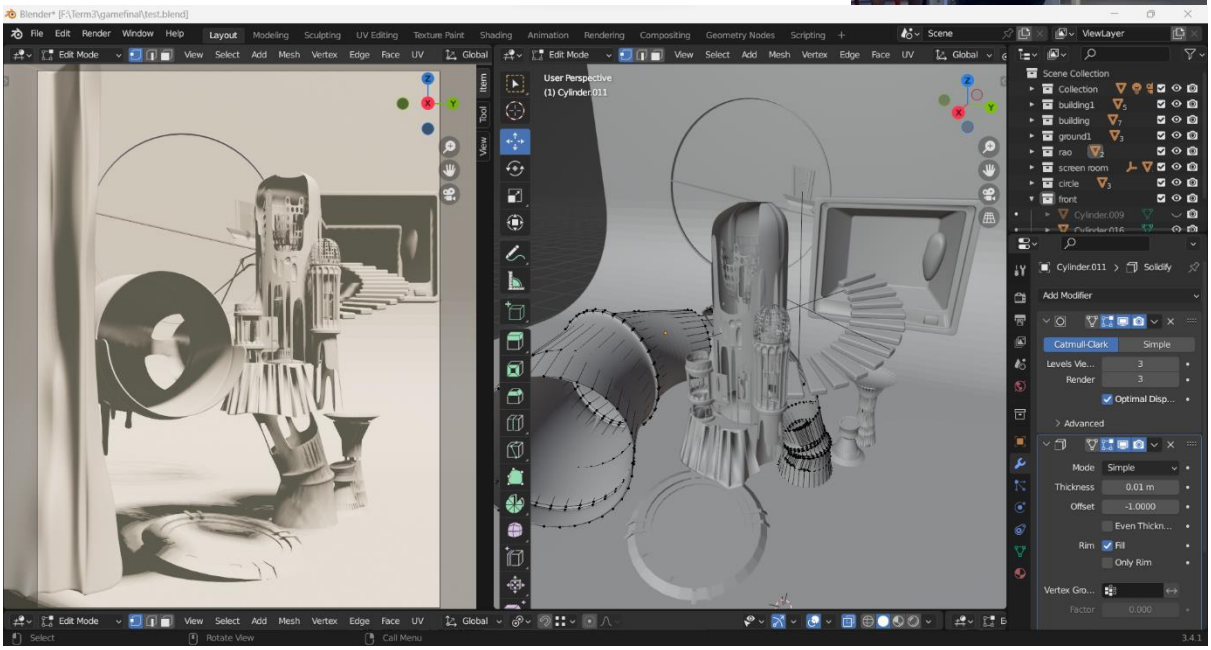
Space

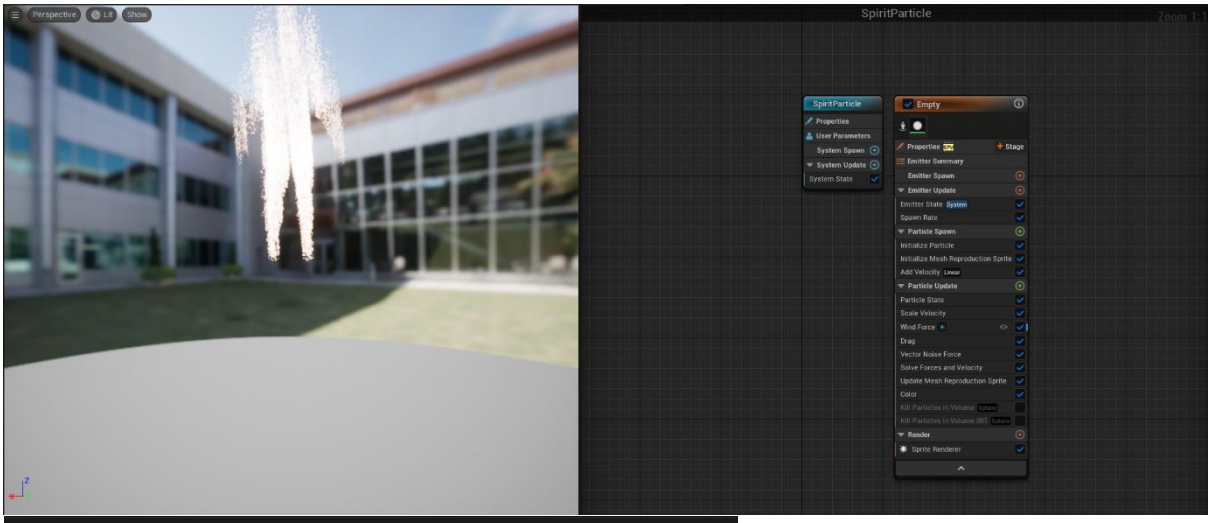


Above: Structure Sketch

Models in Blender & in Real World

We built our buildings in Blender. To differ from the real world, we made our virtual world more dedicated and charming to create a harmony space. We use white texture to make sure our users focus more on the traces and interactions rather than the building itself, but the model is the key point to show the connection between the real world as the model share similar structure with the real world frameworks and structures.



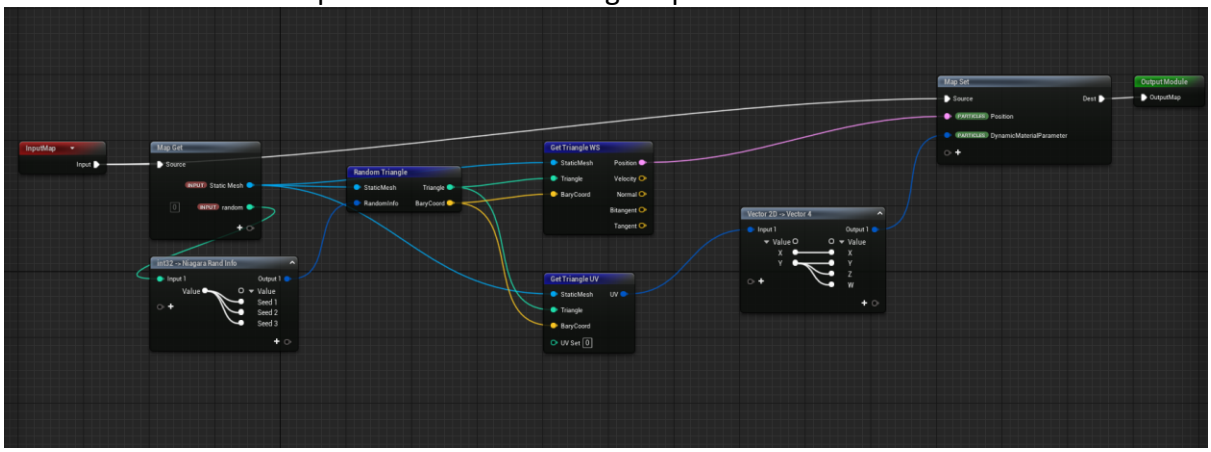


Above is Niagara System used as character figure.

Left is a screenshot when we testing our camera's position in real world to ensure it could track people in the scene.

Models in Unreal Engine

We use a displacement shader graph to overlap the original material, thus create a transparent displacement effect. We placed a ball in the center to indicate the position, but we made it transparent. We also use the displacement map as the particle map, thus particles can emit from the edges of the displaced area and also light up the entire scene.



Shader Graph we use to create displacement map.

Neural Network

For our project, the detection of people from top view is crucial as the camera is the only input that analyze the real-world scene and the position data for further interactions. As I have gained experience on deployment of neural network in game engine via previous projects, we chose to deploy a convolutional neural network as the detection algorithm. To load a neural network in Unreal Engine, we decided to use OpenCV library which can load various types of neural network via its DNN module.

Model - YOLOv5, 6.1ver

https://github.com/ultralytics/yolov5/tree/v6.1

After comparing several computer vision algorithms, I decided to use YOLO model. YOLO is a fully convolutional model using DarkNet as backbone and has been one of the most popular and mature real-time detection models that used for recognition. For a real-time tracing of people from top view, YOLO might be the best and fastest network for us to use.

The reason I did not use more up-to-date model is that we are using OpenCV DNN module to load a ONNX model, there are several operations in latest YOLO model are not supported by either ONNX or OpenCV DNN. *

*Yolov8 is compatible with OpenCV 4.7.0 version. However, for Unreal 4.27, OpenCV 4.7.0 seems incompatible. And Unreal 5 has built-in OpenCV 4.5.5, so I decide to use stable version of OpenCV 4.5.5.

Dataset - Top-view from UCU

https://er.ucu.edu.ua/bitstream/handle/1/1903/Prodaiko_Person%20Re-identification.pdf

https://github.com/ucuapps/top-view-multi-person-tracking

I found the dataset from GitHub, which contains links to multi-person re-identification and tracking dataset in top view multi-camera environment. The dataset contains 5 hours of video from 5 cameras + almost 4000 manually labeled images in YOLO format.

https://github.com/Walpolen/YOLO-V5-Top-view-detect

I had also found this repository that trained YOLOv5 with the dataset, but to convert YOLOv5 into ONNX format that can be load to OpenCV DNN, several changes had to be made. So, I referred to https://blog.csdn.net/nihate/article/details/112731327 (Chinese), and decided to train my own YOLOv5.

```
def forward(self, x):
    z = []  # inference output

    if torch.onnx.is_in_onnx_export():
        for i in range(self.nl):  # 分别对三个输出层处理
            x[i] = self.m[i](x[i])  # conv
            bs, _, ny, nx = x[i].shape  # x(bs,255,20,20) to x(bs,3,20,20,85)
            x[i] = x[i].view(bs, self.na, self.no, ny, nx).permute(0, 1, 3, 4, 2).contiguous()
            y = x[i].sigmoid()
            z.append(y.view(bs, -1, self.no))
        return torch.cat(z, 1)
```

Training

My training repository: https://github.com/Pannic17/G-YOLOv5.6.1

YOLO model was built in PyTorch, so I decided to stick to the original backbone and use PyTorch to train the network.

For the training devices, GPU is NVIDIA GTX2070 and CPU is Intel i7-10700. Due to the limitation of device performance, I have to change several hyperparameters. I reduce the batch-size to 8 and epochs to 10 as I used the pretrained Yolov5s model as initial weights. During training, I also encountered several challenges such as installing correct CUDA/CUDNN version for the GPU and PyTorch, load appropriate dataset size to prevent memory leak, etc.

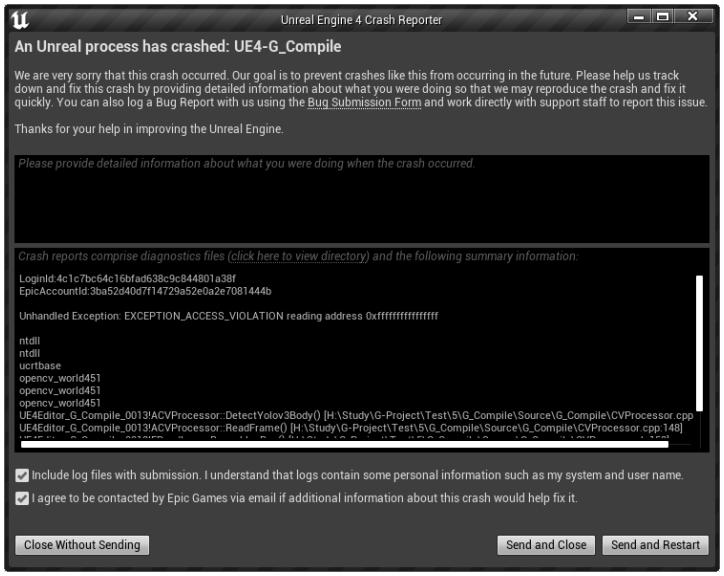
```
parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
parser.add_argument('--data', type=str, default=ROOT / 'data/topview.yaml', help='dataset.yaml path')
parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
parser.add_argument('--epochs', type=int, default=10)
parser.add_argument('--batch-size', type=int, default=8, help='total batch size for all GPUs, -1 for autobatch')
parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
parser.add_argument('--rect', action='store_true', help='rectangular training')
parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
parser.add_argument('--noval', action='store_true', help='only validate final epoch')
parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
parser.add_argument('--cache', type=str, nargs='?', const='ram', help='--cache images in "ram" (default) or "disk"')
parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')
parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')
parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')
```

Deployment - OpenCV DNN for ONNX

ONNX Runtime is an open-source runtime engine developed by Microsoft and other organizations to execute machine learning models in the ONNX format. It offers high-performance inference capabilities and supports a wide range of hardware platforms and frameworks. While, the OpenCV (Open Source Computer Vision) library is a popular open-source computer vision and machine learning library. The DNN (Deep Neural Network) module provides functionalities for working with deep learning models, including inference and pre-trained model integration. So, if I wish to use neural network in Unreal, use OpenCV DNN as intermediate is the best choice.
However, Unreal 4 do not have built-in or free OpenCV third-party library, so I have to integrate OpenCV in Unreal by my own.

I also found a repository using OpenCV library in Unreal.
https://github.com/VegetableWithChicken/OpenCVForUnreal
However, the version used is 4.3.0, which does not compatible with YOLOv5 model.
https://blog.csdn.net/qq_34586921/article/details/111579657
https://blog.csdn.net/nihate/article/details/112731327

Then, I referred to the two articles and managed to deploy OpenCV 4.5.3 into Unreal 4.27 version, but even after I successfully deployed OpenCV, Unreal Engine would crash after several frames and gives a crush report. I tried to solve the error by changing the integration version of OpenCV (tested 4.5.1/4.5.5/4.7.0), but it constantly crashes. After investigating the message, it seems the error is related to the memory allocation issue that caused by UE4 core.



After many attempts, I decided to switch to UE5. UE5 has a built-in OpenCV library available to use, but it also needs some changes in code to be used.

This is a hard decision, because previously our project was built in UE4, especially we used Niagara System, which is quite different in UE4 and UE5. Fortunately, switching from UE4 to UE5 did not take us long time.

Thus, we used YOLOv5 6.1 version model as our neural network. And deployed it in Unreal Engine 5.2 with OpenCV 4.5.5 DNN module. The weights was store in ONNX runtime operation set 12.

Unreal 4 Integration

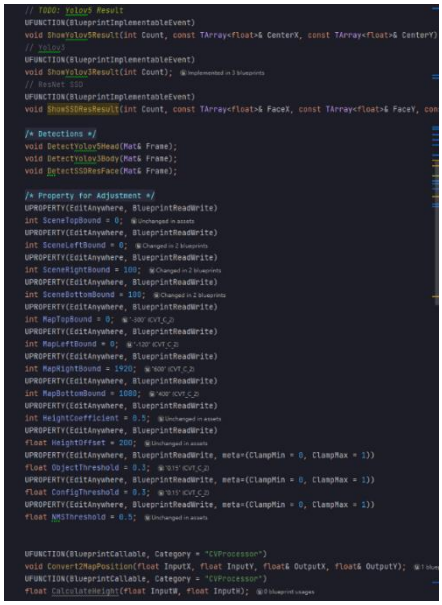
Our goal is to use YOLOv5 to detect person from top-view and convert such coordinate to virtual world location that can drive the character in our UE5 scene to move and interact with the objects.

Custom C++ Code for Neural Network

https://github.com/Pannic17/G-REUE5 (UE5)

https://github.com/Pannic17/G-Mirror-Unreal-OpenCV (UE4)

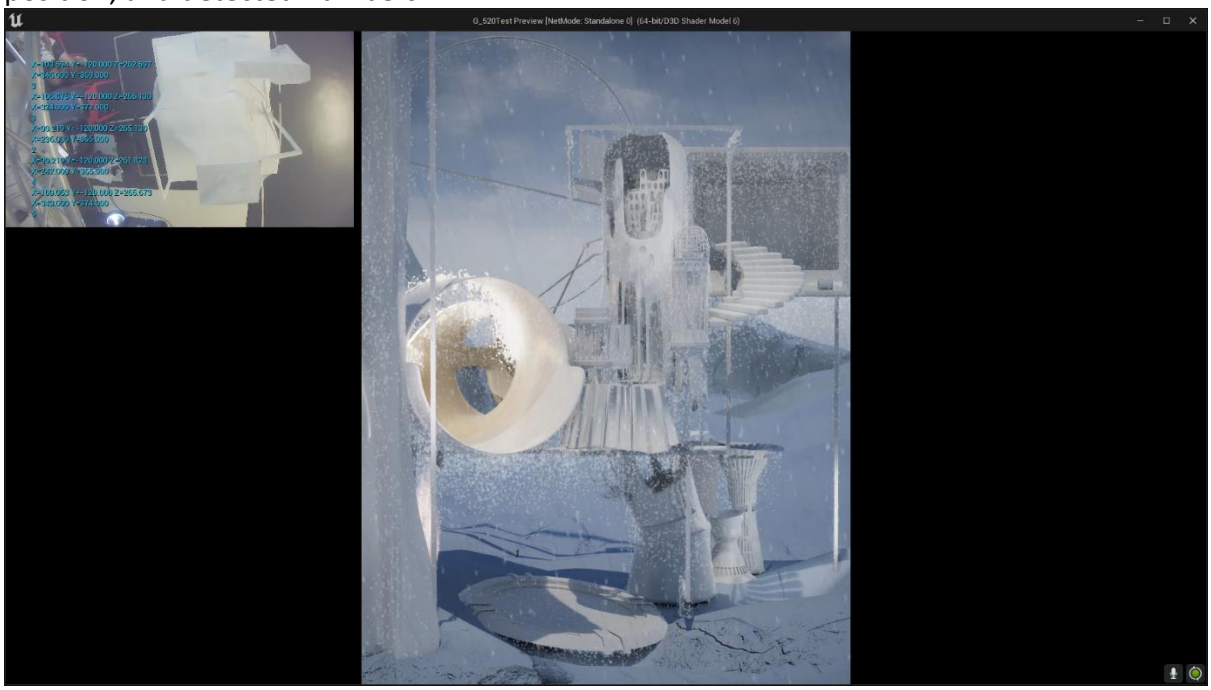
In the Unreal project, how to read and load neural network data is a huge problem. To make the neural network work, I referred to the repository https://github.com/hpc203/yolov5-v6.1-opencv-onnxrun, and replicate the code to make OpenCV in Unreal could successfully do YOLOv5 neural network calculation. For our project, we need get the detection result as the center of the head, the size of the head. Also, for the data available to be used in Unreal, I expose several properties and functions so that these scripts can be used in Unreal Blueprint for further calculation.



Left: Exposed properties and functions
Right: The function used to calculate player position according to detection result

```cpp
void ACVProcessor::Convert2MapPosition(float InputX, float InputY, float &OutputX, float &OutputY)
{
    OutputX = InputX;
    OutputY = InputY;
    if (InputX < SceneLeftBound)
    {
        OutputX = (float) SceneLeftBound;
    } else if (InputX > SceneRightBound)
    {
        OutputX = (float) SceneRightBound;
    }
    if (InputY < SceneTopBound)
    {
        OutputY = (float) SceneTopBound;
    } else if (InputY > SceneBottomBound)
    {
        OutputY = (float) SceneBottomBound;
    }
    float SceneWidth = (float) (SceneRightBound - SceneLeftBound);
    float SceneHeight = (float) (SceneBottomBound - SceneTopBound);
    float MapWidth = (float) (MapRightBound - MapLeftBound);
    float MapHeight = (float) (MapBottomBound - MapTopBound);
    OutputX = (float) (MapLeftBound + (OutputX - SceneLeftBound) / SceneWidth * MapWidth);
    OutputY = (float) (MapTopBound + (OutputY - SceneTopBound) / SceneHeight * MapHeight);
    UE_LOG(LogTemp, Warning, TEXT("X: %f, Y: %f"), OutputX, OutputY);
    // return FVector2D(OutputX, OutputY);
}
```

Except from the neural network calculation and convert detection position to player position, the rest functions are implemented in Unreal Blueprint as its easier to debug rather than re-compile each time I made any changes.
From the screenshot, you can see the top left is a UI viewport of captured images, so that we can aim the camera at the real-world scene, I also logged the detected position, converted position, and detected numbers.



(Screenshot of running)

Unreal Scene

Landscape Interaction - Displacement Map

For landscape interaction, I used Infinity Weather from Unreal Market, which create a snow scene interaction, which is good in UE4. However, in UE5, as the logic of displacement map and tessellation is changed, the effect is not that promising anymore.

UE4 UE5

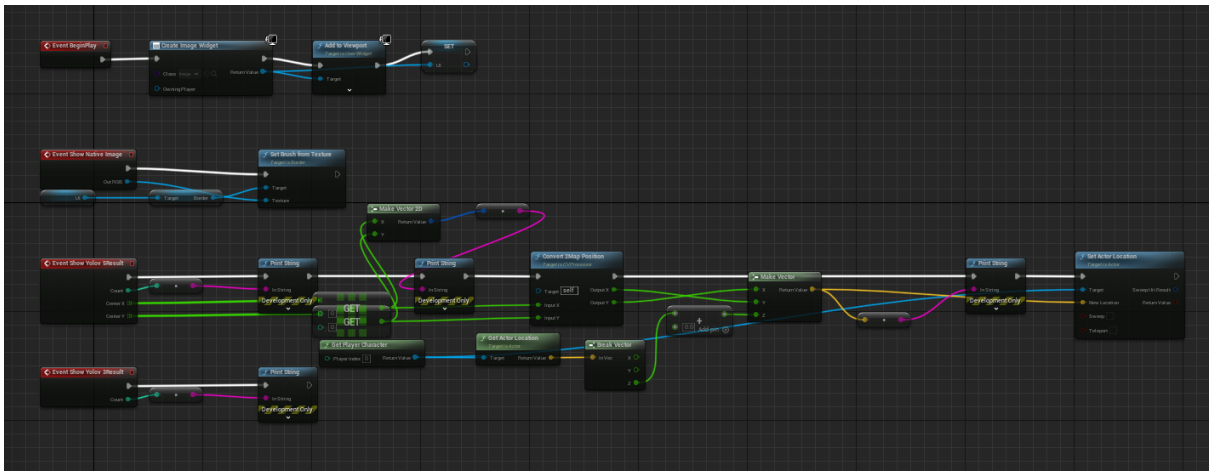
Weather & Interaction – Particles

For the snowing effect, we also used Infinity Weather from Market to create whole scene weather, but we also create our own custom Niagara Particle System that emits from the model.

Player

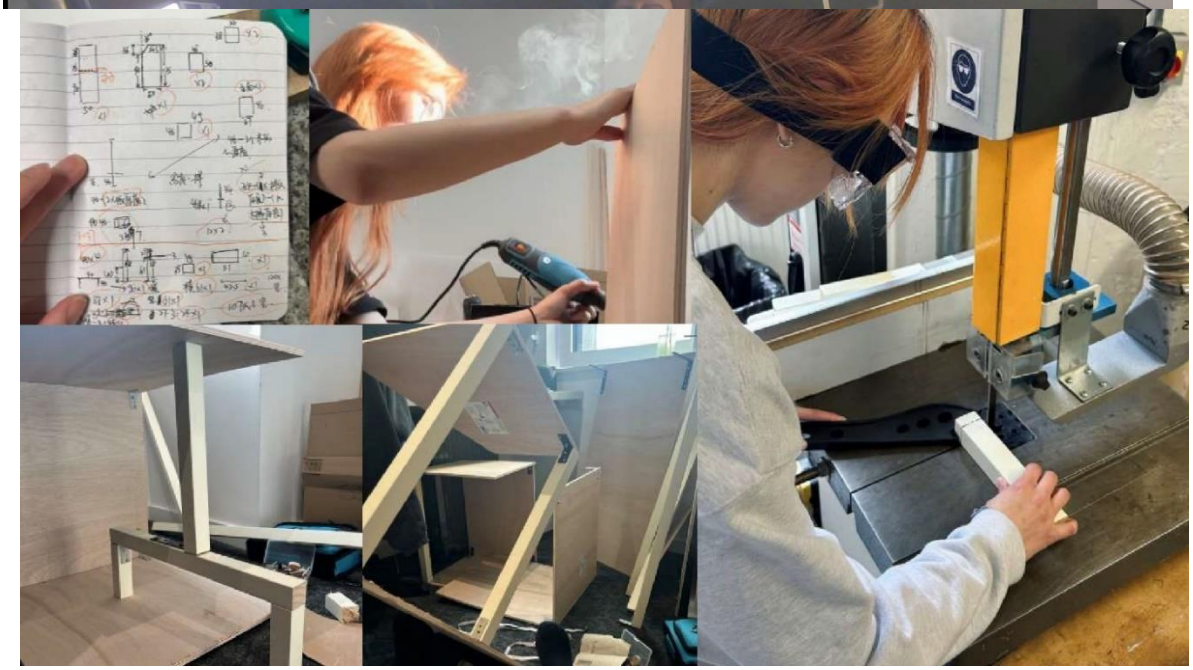
As we focus more on the traces that the player left rather than the player itself, we made our character entirely transparent, but we would also tell our player that you are at the place and show the magic of projecting themselves from real-world to virtual world. So, instead, we use a Niagara Particle System to create a drag afterimage effect to indicate the exact position you are at. The result is fascinating, it seems like you have uploaded your real-world data to a cloud digital world.

we also used a fixed position camera to create a fixed viewport that give our player a feeling of mirror.
To control the player, we used the script I wrote and mentioned to compile a blueprint class that remap the position to a more reasonable coordinate.



Real-World Setup

The real-world aims to create a similarity that make our users feel connected between real world an virtual world. So, we decided to use wood and white paint to reconstruct a structure that player can play with and can also be traced from top view. The site is 1.8m x 1.8m large with a height of 1.35m. We also use white yarn to create a hazy feeling.



Space Design for the Digital World

In the digital world, we have designed a space that corresponds to the virtual realm. Players are given the opportunity to explore this space and immerse themselves in the digital world's wonders. The design of the space aligns with the surrealistic theme, reflecting the erosion of time and incorporating elements such as floating objects, distorted structures, and interconnected elements. It offers a visually captivating and intriguing environment for players to discover.

Designing the Structure

We proposed three different architectural concepts that align with the theme of the digital world. Considering the range of activities, including standing, sitting, and other interactions, we carefully designed the structure to accommodate these movements. Ultimately, we constructed a space that allows players to freely navigate and engage within it. The architecture embodies the surreal elements of the digital world, forming a harmonious blend of aesthetics and functionality.

Prototype

The project was adapted from the project we did last term for the exhibition. The project we did last term, Anybody Problem, uses a ResNet SSD model in UE4 via OpenCV DNN third-party library, and used the detected human face as gravity point for Niagara System.

References

https://github.com/ultralytics/yolov5/tree/v6.1
https://er.ucu.edu.ua/bitstream/handle/1/1903/Prodaiko_Person%20Re-identification.pdf
https://github.com/ucuapps/top-view-multi-person-tracking
https://github.com/VegetableWithChicken/OpenCVForUnreal
https://github.com/hpc203/yolov5-dnn-cpp-python
https://github.com/hpc203/yolov5-v6.1-opencv-onnxrun
https://blog.csdn.net/nihate/article/details/112731327
https://blog.csdn.net/qq_34586921/article/details/111579657
https://www.youtube.com/watch?v=YOidIl2kTD0
https://github.com/Dukhart/UE4_OpenCV
https://github.com/Walpolen/YOLO-V5-Top-view-detect