Ain Shams University
Faculty Of Engineering
Architecture Department
Post Graduate Studies

Integrating Game Technology in Architectural Visualization

A Research submitted in Partial Fulfillment
Of requirements for the degree of
Master of Engineering in Architecture

Presented by Architect
Ahmed Mohamed Yousry Lashin
B.Sc. Architecture - Ain Shams University, Class 2008

Supervised By
Dr. Hazem Talaat El-Daly
Assistant Professor
Department of Architecture
Faculty of Engineering
Ain Shams University

Cairo
Egypt
January 2016

Table of Contents:

List of Figures:

Abstract:

Many Architects nowadays use Game Engines as a presentation tool; thus using its interactive capabilities to present the product and interact with the clients.
A growing movement among some architects to use the photo realistic power of game engines for visualization purposes. That need, combined with the game engines that are available on the market today, is what created the interest in furthering this research, finding new methods to represent the product to the clients.

Introduction:

Through the years, the continuous search & development of new software has never stopped, every day, there is more and more in the computer aided design field, but that didn't prevent the reuse of other software to serve as a design & Presentation tool; Discovering new potentials in the architectural design presentation process made by these programs.

Research Problem:

Current methods to display a new product in the architectural visualization industry involve long render times and hundreds of frames that require rendering. Many times, these virtual tours that are produced are slow, methodical, and limit the viewer's perspective of the product .still presented the traditional way by a direct interaction with the clients.

Research Goal:

This research looks into using computer game engines to display the virtual tour in real time, thus removing the long render time requirements and limited viewer perspective, and the ability to publish and represent the model online to interact with the client from anywhere any time.

Research Objective:

- Investigating visual presentation techniques and environmental elements that can help and guide users of interactive virtual worlds using game engines.

Research Motivation:

1- No rendering time, quick to publish.
2- Gives viewers freedom to navigate to any place, look at from any viewpoint.
3- It is online. The project could be published to the World Wide Web and communicate with others.
Compared to the expensive high-end VR lab’s technology, game engine is more practical for most general founded researchers and architecture institutes.

Research Scope & Limitations:

- The study is concerned with: presenting an architecture design online using game engines as presentation tool.
- the thesis is not studying nor analyzing common used Architecture software, the thesis just studying the main idea of these tools along with the Virtual online world compared with game engines in visualizing Architecture design.
- The study is also concerned with Virtual Reality through Games as an application on it , Games are being created with various software that is not in this thesis scope of work ,we are studying the results of using this software called *game engines* in the Building design visualization process especially when all games nowadays use architecture as it's playground.
- In this thesis we are focusing on Microsoft Windows platform.
- Our target group is architects.
- The research is limited by the type and quantity of data and information sources available for the subject and it depend mainly on published literature; books, scientific papers, researches and conferences proceeding.
- Case study project that show features of applying interactivity within its context.

Research Structure:

Fig. 1: Research Contents

1. Games:

1.1. Introduction

Games provide an ideal environment in which to study computational intelligence, offering a range of challenging and engaging problems. Game theory captures the behavior in which a player's success in selecting strategies depends on the choices of other players. [1]

1.2. Game Definition

Game is a formal model of an interactive situation. It typically involves several players; a game with only one player is usually called a decision problem. The formal definition lays out the players, their preferences, their information, and the strategic actions available to them, and how these influence the outcome[2].

A. Games environment:

The explanation of this class is exactly the causes that many prefer to play a game rather than to have a lesson. No doubt that many entertainment activities may cause engagement, but gaming is the most activity that engages the players who involve in because:

- Games are an interactive dynamic form of human activity.
- Gaming is a funny challenging activity.
- Games always have an aim, "To Win".
- Games in most cases are flexible.

B. Games factors:

There are the three types of entertainment activities; static media, dynamic media and interactive media.
Static media is a snapshot of reality frozen in time. For example; a photo, a landmark or a painting are static media.
Dynamic media show change with time which make them able to represent the changing aspect of reality more rich. Like movies, music, dance, and other similar Medias.
Interactive media represents how things react towards one's own action through time.
Why a game can be defined as an interactive media, because certain components are contained within the activity. Any game should contain six factors to be considered as a game:

- Rules & conditions.
- Goals and Objectives.
- Conflict, Competition, Challenge.
- Opposition.
- Interaction.
- Story or theme.

[1] Game Engines for Architectural Visualization in Design, Kevin R. Conway
[2] Theodore L. Turocy , Bernhard von Stengel , Game Theory - CDAM Research Report LSE-CDAM-2001-09 ,October 8, 2001

1.3. Video Games making

Video games are about a challenge in a certain environment, this environment is the game architecture. The game itself is a combination of characters and objects with scripts added to them, and the interface that the player will navigate the game through it.
As for the environment , a lot of architectural elements has been used in order to achieve the game purpose, 2d games dominated the scene at first, but due to modern technology, 3d games has become the first choice of companies and players, because it is closer to reality.
3D games nowadays has multiple environments; urban scale, 3rd person shooter (which is used in this research), 1st person shooter, sports.
Finally, most video games present their stories and respond to player input in real-time, making them interactive real-time simulations. One notable exception is in the category of turn-based games like computerized chess or non-real-time strategy games. But even these types of games usually provide the user with some form of real-time graphical user interface[3].

Fig. 2: 2D Game - Mario - Nintendo


Fig. 3: 1st Person Shooter - Crysis - Electronic Arts


Fig. 4: 3rd Person Shooter –The Division

1.4. Game Engine

The term "game engine" was originally founded in 1990s in reference to first-person Shooter (FPS) games like the insanely popular Doom by id Software.
A game engine is a software framework designed for the creation and development of video games. Developers use them to create games for multiple consoles.
As game engine technology matures and becomes more user-friendly, the application of game engines has broadened in scope. They are now being used for serious games: visualization, training, medical, and military simulation applications, with the CryEngine being one example[4].
A data-driven architecture is what differentiates a game engine from a piece of software that is a game but not an engine. That means when a game contains hard-coded logic or game rules, or employs special-case code to render specific types of game objects, it becomes difficult or impossible to reuse that software to make a different game[5].

Fig. 5: Common used Game Engines

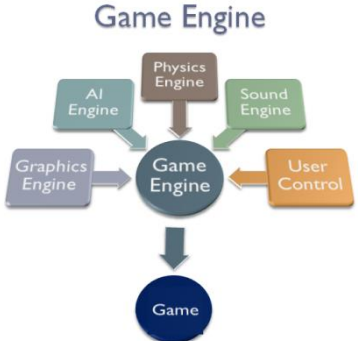


Fig. 6: How game engine works

[3] Game Engine Architecture - Jason Gregory P.8
[4] Video Games Starting to Get Serious. Gazette.net. 2007-08-31. Retrieved 2011-01-17
[5] E. Haque, Mohammed , Dasgupta, Pallab ; Architectural/Engineering Visualization using Game Engine ; Proceedings of the 2008 ASEE Gulf-Southwest Annual Conference , The University of New Mexico – Albuquerque.

1.5. Unity Game Engine:

The Unity game engine is one of the best, common used video game engines. The first time it saw the light was by Unity Technologies in 2004 as a development tool; it was later launched in 2005 at Apple's Worldwide Developers Conference[6]. Now it is being used in more than just creating games & entertainment, it has been involved in other fields such as architectural project presentation as this research will discuss.

Fig.7: Unity Logo

1.5.1. The Ease of the Unity Editor:

Unity engine is very easy to use learn & use. Due to it's simple and logic interface; This engine focuses a lot on simplifying the game development workflow, and nowhere is that more apparent than with the Unity Editor. The Unity editor even goes so far as to run the game in the game window so that you can see a preview of how it will look on your target device.

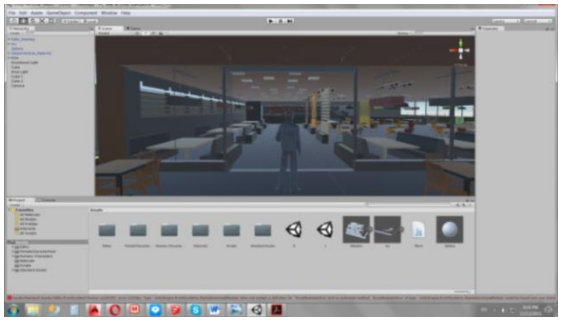


Fig.8: Unity Interface

The ability to run your game while simultaneously seeing the properties and locations of all objects in the scene is a very powerful and time saving feature[7], it allows the user to determine the drawbacks in his progress while testing the game and evaluating the results.

1.5.2. Scripting:

In unity engine, two programing languages have been built in the software, which gives the ability to add any written script to a character or an object in the scene. With mono editor, it is easy to add multiple scripts and codes for the objects. Unity allows for powerful behaviors written in any of two languages: JavaScript, C#. Furthermore, the two languages can be used at the same time within a project to allow people of different technology backgrounds to contribute to a project at the same time[8]. Also each error occurs has it's own code which developer can search on to learn and fix it before running the game.

1.5.3. One Source:

A very important feature of Unity engine is the ability to build projects for multiple platforms with just a click. Select the Platform and Build, Unity can build for Windows PC, Linux (new with version 4), iOS (with plugin), Mac, Android (with plugin), Web Browser, Flash (with plugin), PS3, Xbox, and Wii[9].

1.5.4. Pricing and Licenses:

Unity has it's own online store for assets purchasing. This means that the developer can download any assets he needs without wasting his time creating new ones[10].

Fig. 9: Play Maker Page

[6] How Unity3D Became a Game-Development Beast. Slashdot.org. Dice. June 3, 2013. Retrieved July 13, 2014.
[7] Campbell, Dace A. ; A Critique of Virtual Reality in the Architectural Design Process , University of Washington ,Seattle, WA 91895
[8] How Unity3D Became a Game-Development Beast". Slashdot.org. Dice. June 3, 2013. Retrieved July 13, 2014.
[9] D. Chao, "Doom as an interface for process management," in Proceedings of SIGCHI'01,pp. 152

2. Architecture in Games:

2.1. The Primary Function of Architecture in Games

The primary function of architecture in games is to support the game play. Buildings in games are not analogous to buildings in the real world, because most of the time their real-world functions are either irrelevant - the real-world activity that the building serves isn't meaningful in the game- or purely metaphorical. Rather, buildings in games are analogous to movie sets: incomplete, false fronts whose function is to support the narrative of the movie. Movie sets create context and support suspension of disbelief. They also diverge from the real world for narrative purposes[11].

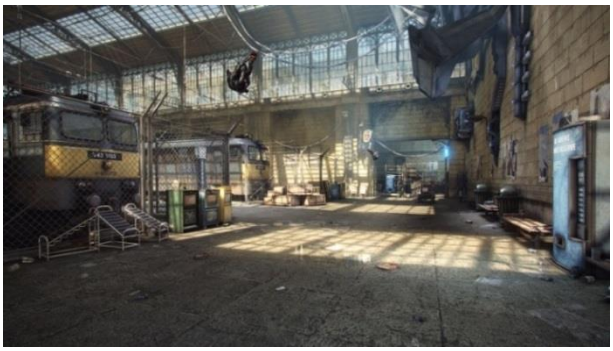


Fig. 10: Architecture in Games

Constraint: Architecture establishes boundaries that limit the freedom of movement of avatars or units. It also establishes constraints on the influence of weapons. As a general rule, projectiles do not pass through walls nor do explosions knock them down, nor fires burn through them.

Concealment: Architecture is used to hide valuable objects from the player; it's also used to conceal the players from one another, or from their enemies.

Obstacles: obstacles & traps must be made as a part of the landscape architecture of computer games. Some of them can be surmounted by observation and logic, others by hand-eye coordination.

Fig. 11: Prince of Persia - Ubisoft

Exploration: Exploration challenges the player to understand the shape of the space he's moving through, to know what leads to where. Mazes are of course one of the oldest examples of such a challenge. If the game doesn't give the player a map, he may have to rely on his memory to learn his way around.

[10] Campbell, Dace A. ; A Critique of Virtual Reality in the Architectural Design Process , University of Washington ,Seattle, WA 91895, pp 122

[11] A. Herwig and P. Paar, Trends in GIS and Virtualization in Environmental Planning and Design, ch. Game Engines: Tools for Landscape Visualization and Planning?, pp. 161–172. Wichmann Verlag, Heidelberg, 2002.

2.2. The Secondary Function of Architecture in Games

If architecture were only about supporting the gameplay through constraint, concealment and so on, it could all be bare grey concrete. But architecture has a secondary and still highly valuable role to play: to inform and entertain in its own right. It does this by a variety of means:

Familiarity: Familiar locations offer cues to a place's function and the events that are likely to take place there. We rely on players to use common sense about the function of certain kinds of familiar spaces, and it's cheating (a conceptual non-sequitur) to violate their legitimate expectations without any explanation. If you can crawl through the ventilation ducts to get past the security guards, it's not reasonable to meet another security guard inside the ducts.

Fig. 12: Hitman2 – io interactive
Inside petronas towers

Allusion: Game architecture can make reference to real buildings or architectural styles to take advantage of the ideas or emotions that they suggest. There's a vast amount of material to borrow from in the real world, from the ruinous spiritual grandeur of Stonehenge to the gruesome expediency of the gas chamber at San Quentin.

Surrealism: It creates a sense of mystery and more importantly, it warns the player that things are not what they seem. A surreal landscape tells him that the game may require extreme lateral thinking or strange leaps of logic to win.

Fig. 13: Metro Last Night – 4A Games

Fig. 14: Fifa 2016 - EA Games

3. The Project Presentation Process:

Developing a virtual 3D environment by using game engine is a strategy to incorporate various multimedia data into one platform. The characteristic of game engine that is preinstalled with interactive and navigation tools allows users to explore and engage with the game objects. However, most CAD and GIS applications are not equipped with 3D tools and navigation systems intended to the user experience. In particular, 3D game engines provide standard 3D navigation tools as well as any programmable view to create engaging navigation thorough the virtual environment. By using a game engine, it is possible to create other interaction such as object manipulation, non playing character interaction with player and/or environment.

3.1. The Project:

The research focus on using unity game engine as a project presentation tool, so to show how the process works and how the outcomes will be recorded, an architectural project will be chosen that is being developed by the researcher.

It is important to apply the game engine on a real project so every step and problem that interrupts the whole process and the solutions for it can be recorded, by doing that, the experiment will make the research concept real and reliable.

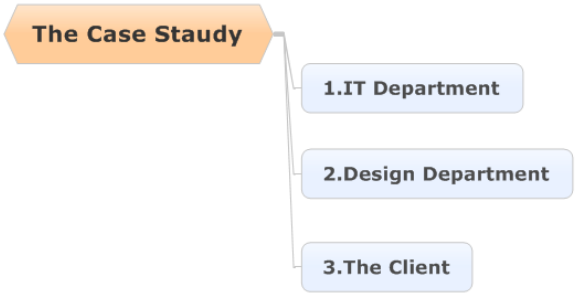
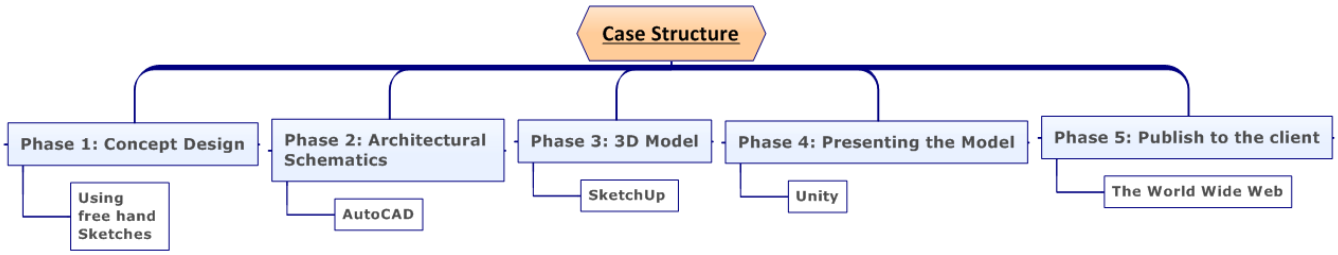


Fig. 15: The Involved Members

The process will start from obtaining the Unity game engine software along with it's assets by the I.T. staff, and then it will be used in order to present the pre- modeled project. Finally, the presentation will be exported as a web game in order to present it to the client, and get the feedback on the proposal from him.

3.1.1. Project Development



3.1.2. Project selection

The project is needed to have the following:

1- Outdoor & Indoor Areas.
2- Multiple stories (minimum 2).
3- Different openings types (doors & windows).
4- Different circulation paths.

3.2. Project Identification

A drive-through, or drive-thru, is a type of service provided by a business that allows customers to purchase products without leaving their cars. The format was pioneered in the United States in the 1930s by Jordan Martin, but has since spread to other countries. Orders are generally placed using a microphone and picked up in person at the window.

3.2.1. The Client:

McDonald's is the world's largest chain of hamburger fast food restaurants, serving around 68 million customers daily in 119 countries across 35,000 outlets. Founded in the United States in 1940, the company began as a barbecue restaurant operated by Richard and Maurice McDonald.

Fig. 16: Mcdonald's Logo

3.2.2. The Project:

The project is a 2 stories fast food restaurant; it is required to house 600 visitors along with 18 staff members, it consists of:

1- Ground floor.
2- 1st floor.
3- Outdoor area.
4- Parking.
5- A drive thru lane.
6- A ring road around the building.

The foot print area of this building = 1530m2.

Land area = 4900m2.

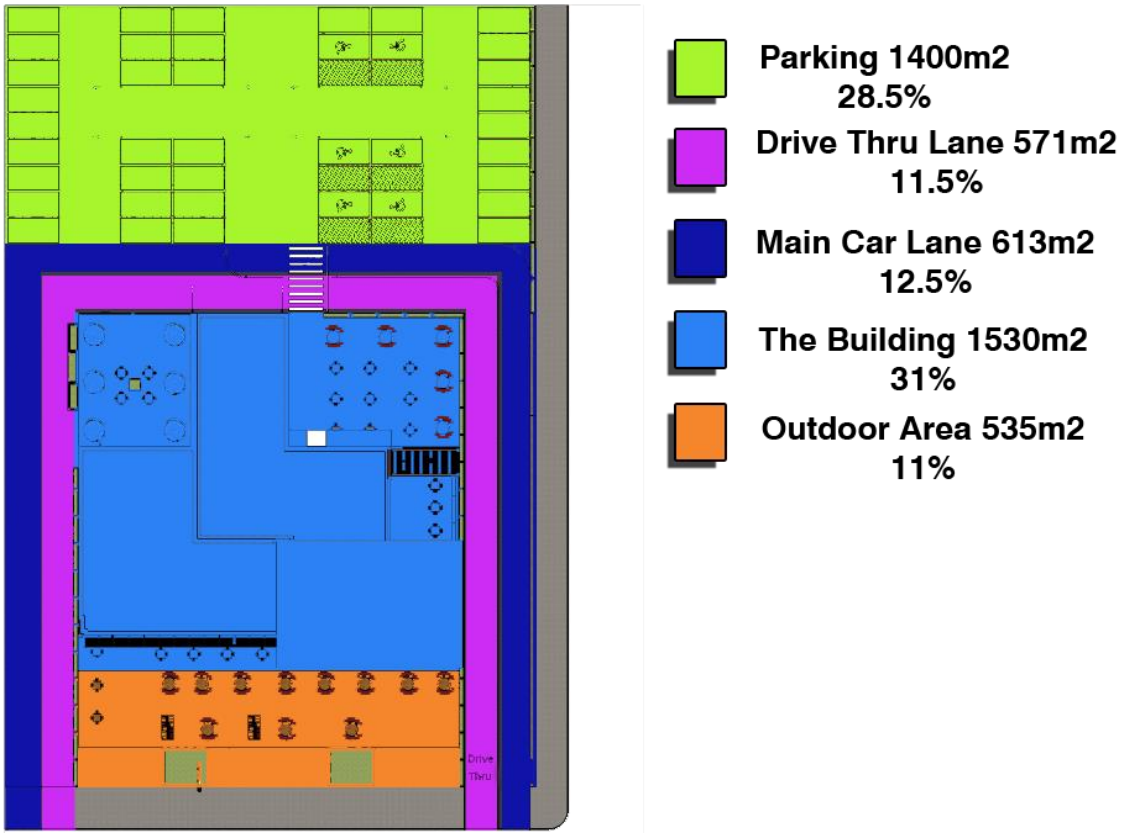


Fig. 17: Layout zoning

3.2.3. The Project Location:

The Project is located at 90th street – New Cairo, in front of the AUC and next to American plaza and the future university in Egypt.

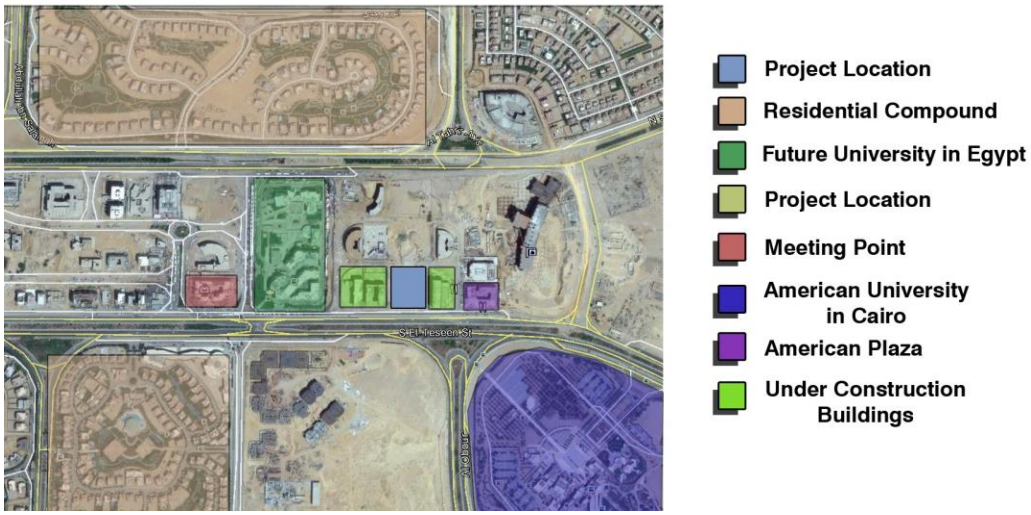


Fig. 18: Layout analysis

Fig. 19: The Project Layout

A picture shows how the building is going to take place in reality.

Fig. 20: Picture of the project location

Fig. 21: the project in the site

The project is located between two under construction buildings, directly on the 90th axis that is splitting new Cairo.

The picture shows how the building will be located in the layout along with the other buildings that already exists.

Fig. 22: Project Layout in site

3.2.4. The Project Schematics:

1. The Ground Floor:

On a 1530m2, the floor consists of:

A- A main seating indoor hall with 500m2.
B- A café for special drinks & beverages 260m2.
C- Kitchen 160m2.
D- Service counter for the indoor area 100m2.
E- Service counter for the drive thru users 20m2.
F- Storage 90m2.
G- Toilets 80m2.
H- Group's area 230m2.
I- Outdoor Terrace 420m2.

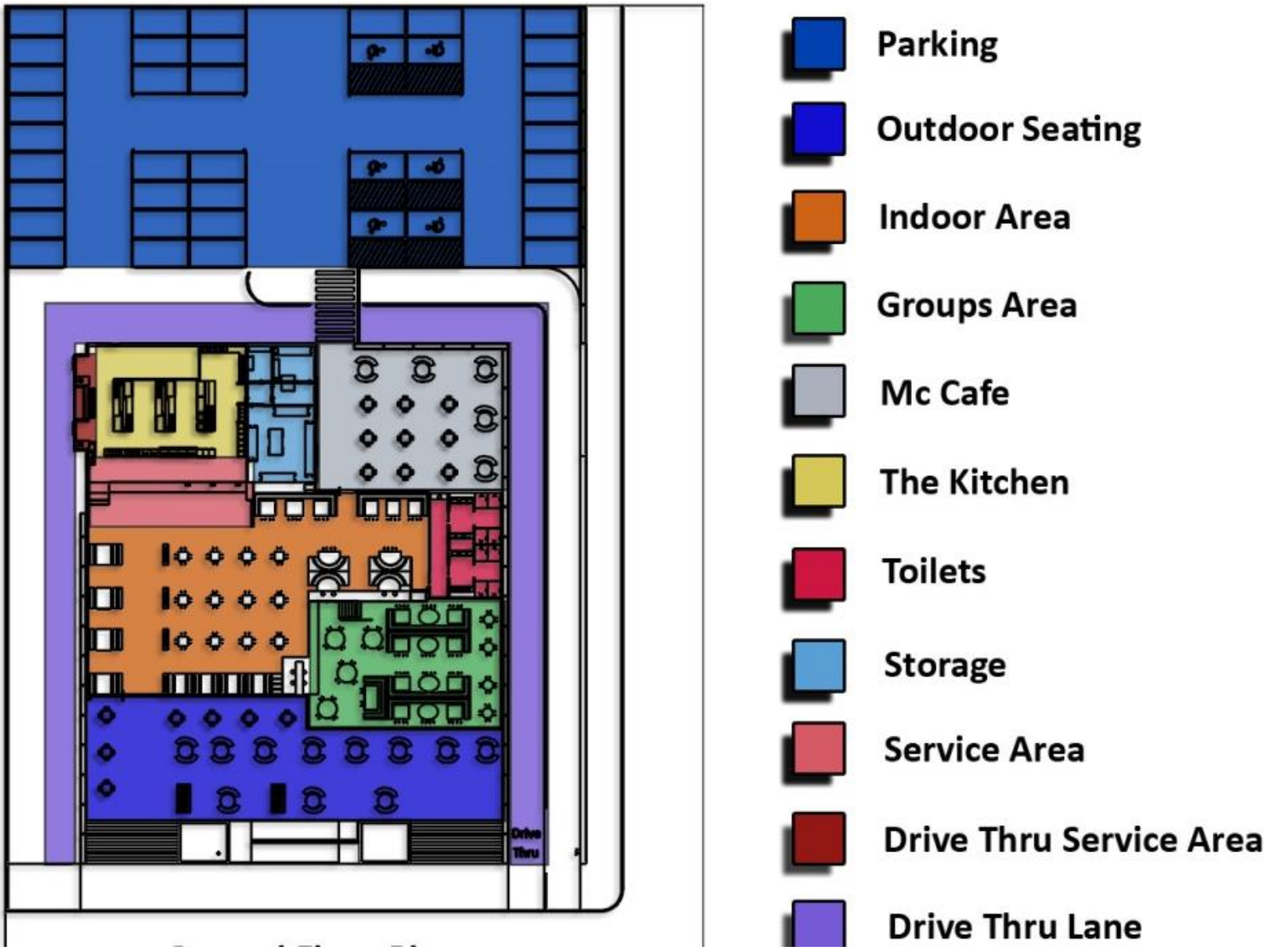


Fig. 23: Ground Floor Plan

2. The First Floor:

On a 990m2, the floor consists of:

A- Seating hall 670m2.
B- Restaurant terrace 170m2.
C- Café terrace 75m2.
D- Toilets 75m2.

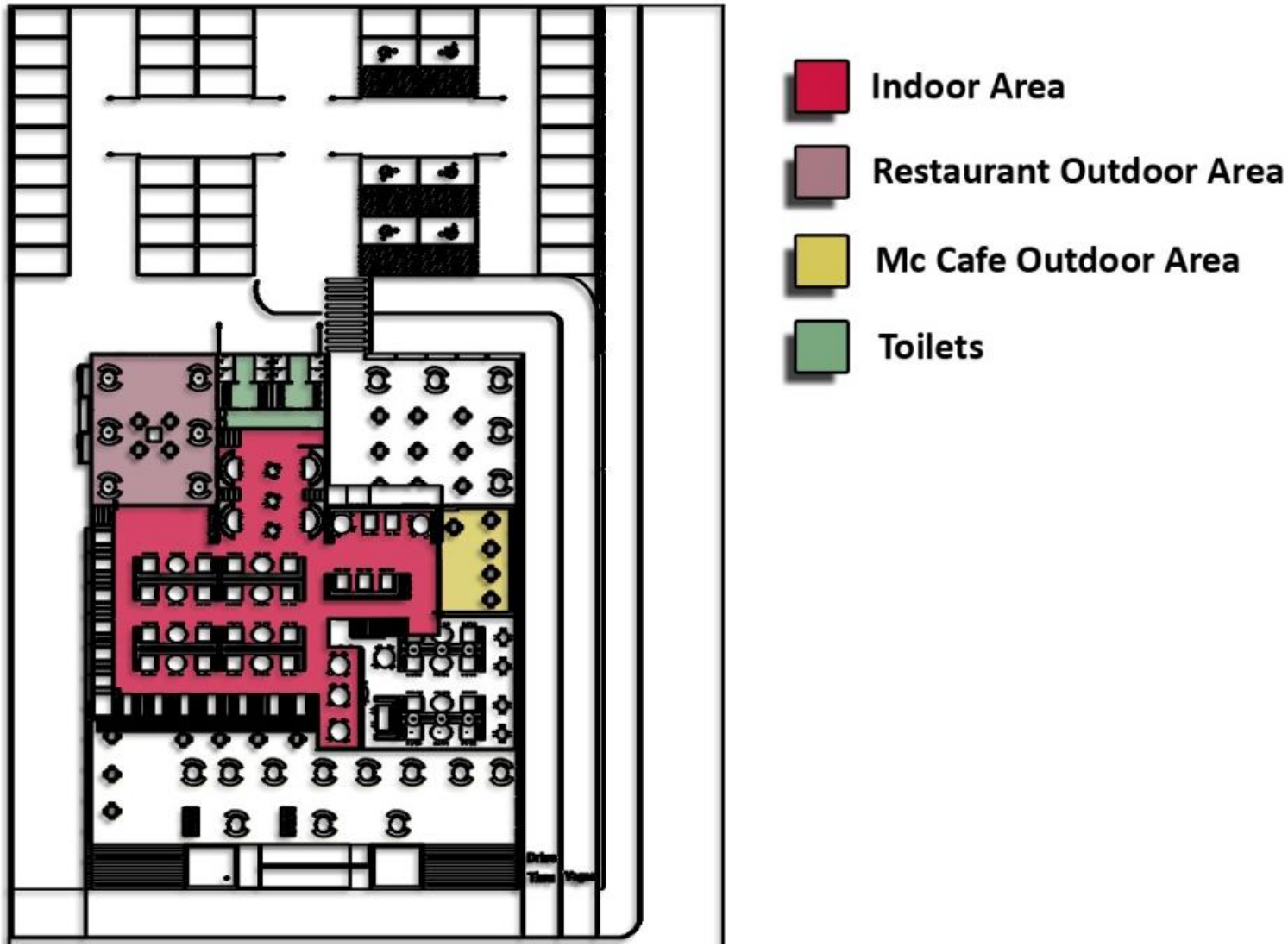


Fig. 24: First Floor Plan

4. The Process:

4.1. Phase 1: The Project Sketch of the Design Concept:

Most architects and designers start the process of design by sketching. Quick and frequently scribbles often define the concept and direction of a project.

These quick and often highly symbolic doodles are a critical first step in the design process. Until the sketch is created, a concept is a hazy, perishable possibility of a design; the sketch is putting the idea on paper.

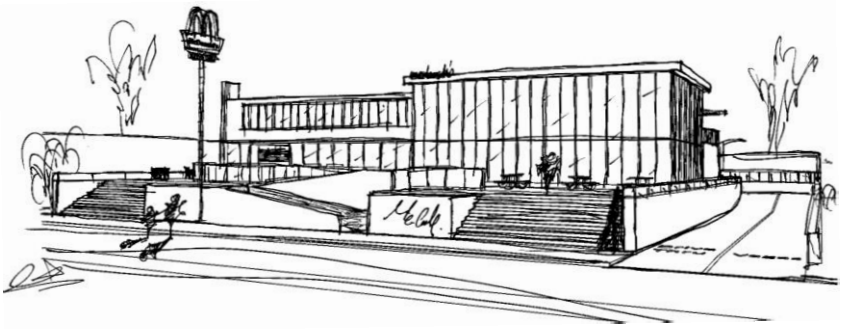


Fig. 25: Sketch of the project

4.2. Phase 2: Architectural Schematics:

After finishing the concept and deciding the main design lines of the project, it is time for the project to enter a to scale stage, using AutoDesk product: AutoCAD Architecture, the schematics will be drawn as required to achieve the client's needs.

The 2D process:

- Starting the project.
- Zoning.
- Drawing the spaces of the project.
- Completing the floor plans.
- Drawing the elevations.

Drawing with AutoCAD is not just about getting it done. In the design process, the architect will need to modify his drawings. He may also needs to send the drawings to contractors or partners, so they can do work based on your drawing. You need to make sure you can make modifications easily and your partners can easily use your drawings.

2D elevations are created by drawing an elevation line in front of a number of objects and then creating a 2D elevation object from them. Editing a 2D elevation is done by changing its object display properties or its style display properties. The 2D elevation style lets us add display components to the display representation of the elevation and create rules that assign different parts of the elevation to different display components.

Drawing is part of the design process. Means it will be needed to create, modify, and share drawings. It is important to make it comfortable for everyone to work with the drawings, as it is used as a reference in the following stages of the process.

4.3. Phase 3: 3D Modeling

Now after the architectural schematics have been made, it is time to move on to the next level; the 3D model. Using SketchUp software, the architecture drawing's files will be imported into SketchUp. Then it will be extruded to form the mass.

In an imported CAD file, SketchUp automatically discards any entities that have no 3D relevance, such as text, dimensions, hatching, logos, and so on. However, SketchUp won't discard the layers holding these entities.

In SketchUp, geometry that's many miles or kilometers away from the origin (0,0) can cause performance problems. This must be avoided by checking the placement of geometry in CAD file.

Generally, CAD files import into SketchUp successfully when the file size is 20MB or less. When importing larger CAD files, the import can take a long time or may fail. Conversely, the smaller your CAD file size, the quicker and easier the import.

Import only the necessary geometry. SketchUp models can be designed to be as accurate as models in CAD. However, SketchUp is not designed for the same type of line-intensive drawings done in CAD software. In the CAD file, cleaning up any content that is not needed to use after we import the CAD file into SketchUp Pro. Simplify the CAD file to just walls, doors and windows.

Separate levels of detail among different CAD files. If our CAD file holds lots of necessary geometry, consider whether we break one file into a few smaller files. For example, one imported CAD file can contain site plan information, another can have a floor plan, and a final file can have a specific detail.

The file units are very important to know whether using inches, feet, or a metric unit of measurement. That way, we can match the SketchUp model's units to the CAD file's units and thus maintain the scale and dimensions of our imported CAD geometry.

In SketchUp Pro, open the SketchUp model into which we want to import our .dwg or .dxf file.

1. Select File > Import.
2. Navigate the place where CAD file is saved.
3. From the Files of Type drop-down list, select AutoCAD Files (*.dwg, *.dxf).
4. Then select the file to import.

Draw a rectangle from the end of one of the wall corners. We notice that the unfilled polyline area also fills in. Erase the new rectangle and its related lines.

With the push-pull tool the walls can be extruded to the desired height.

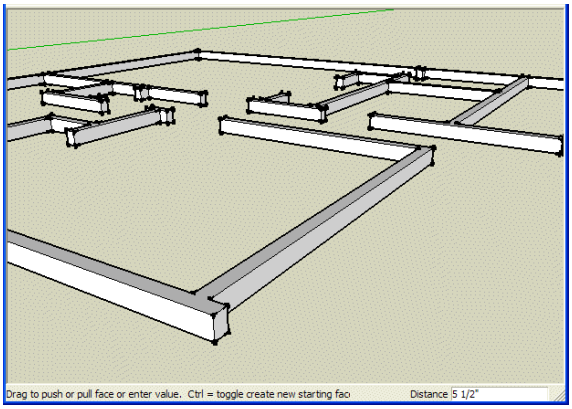


Fig. 26: Extruding Walls in SketchUp

After extruding walls, the openings must be made as per the design; solid and void areas will be determined depending on the earlier drawn schematics; and then assigned to the solid mass.

Now, we will have a mass with only openings, to bring it to life, it must be fully furnished.

From Google SketchUp warehouse, the furniture can be downloaded, then the furniture will be put to the scene each item in it's place. We will use the cad drawings as our reference.

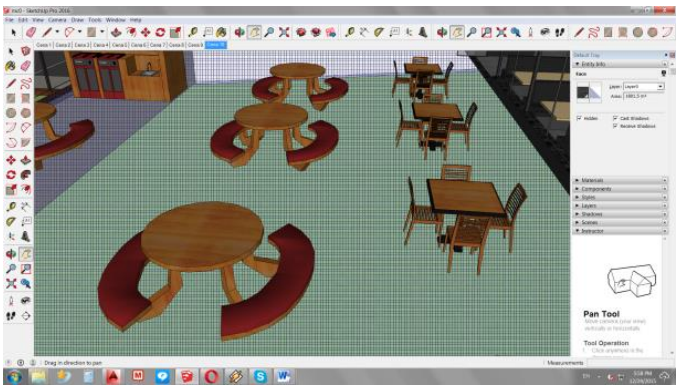


Fig. 27: Furniture in SketchUp

To add detail and realism to our model, SketchUp enables us to paint materials on faces. Materials are essentially paints that have a color and optional texture.

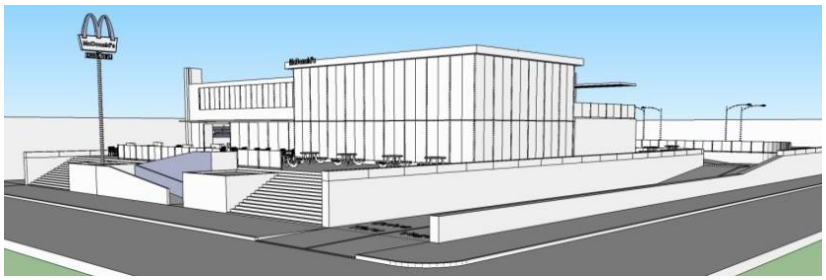


Fig. 28: The Model Without Materials

After we apply materials like these to our model, the special capabilities of SketchUp's materials can help us do any of the following:

Fig. 29: The Model with the Materials

Edit the material: Because the color and texture are separate, we can change them independently of each other. For example, we can change the siding color, but keep the same texture. We can also edit a material's opacity, which controls how opaque or transparent the material is.

Fig. 30: 3D model: Bird eye View 1

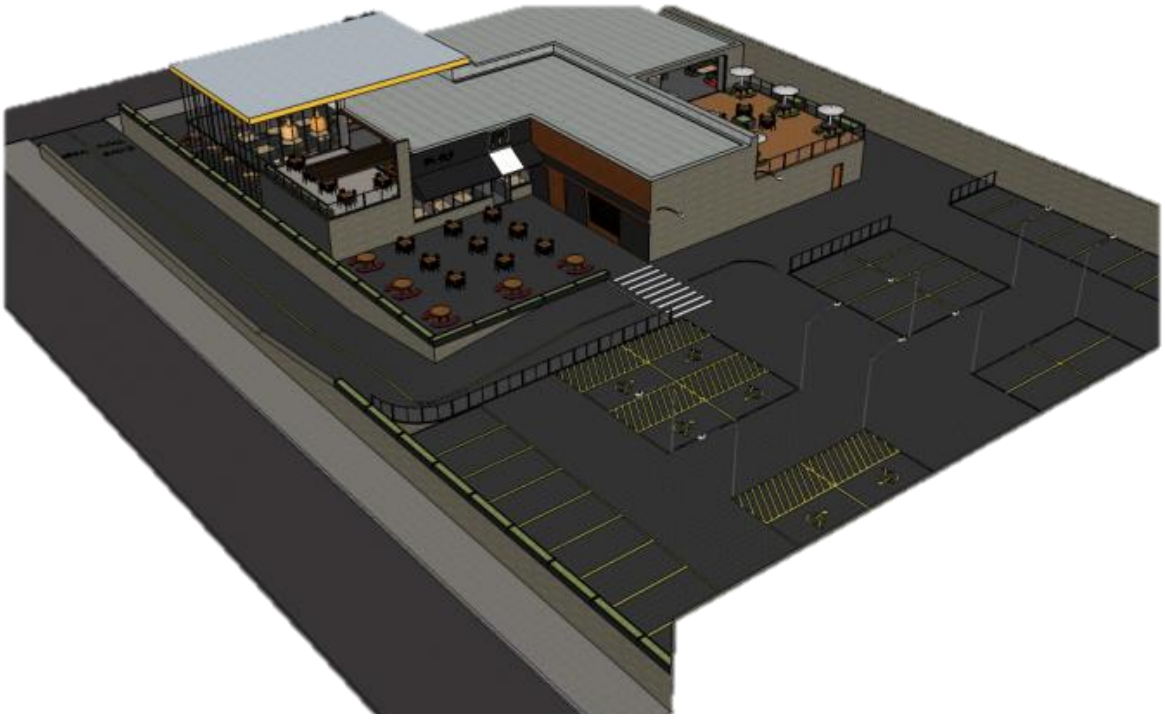


Fig. 31: 3D model: Bird eye View 2

Fig. 32: 3D model: Bird eye View 3

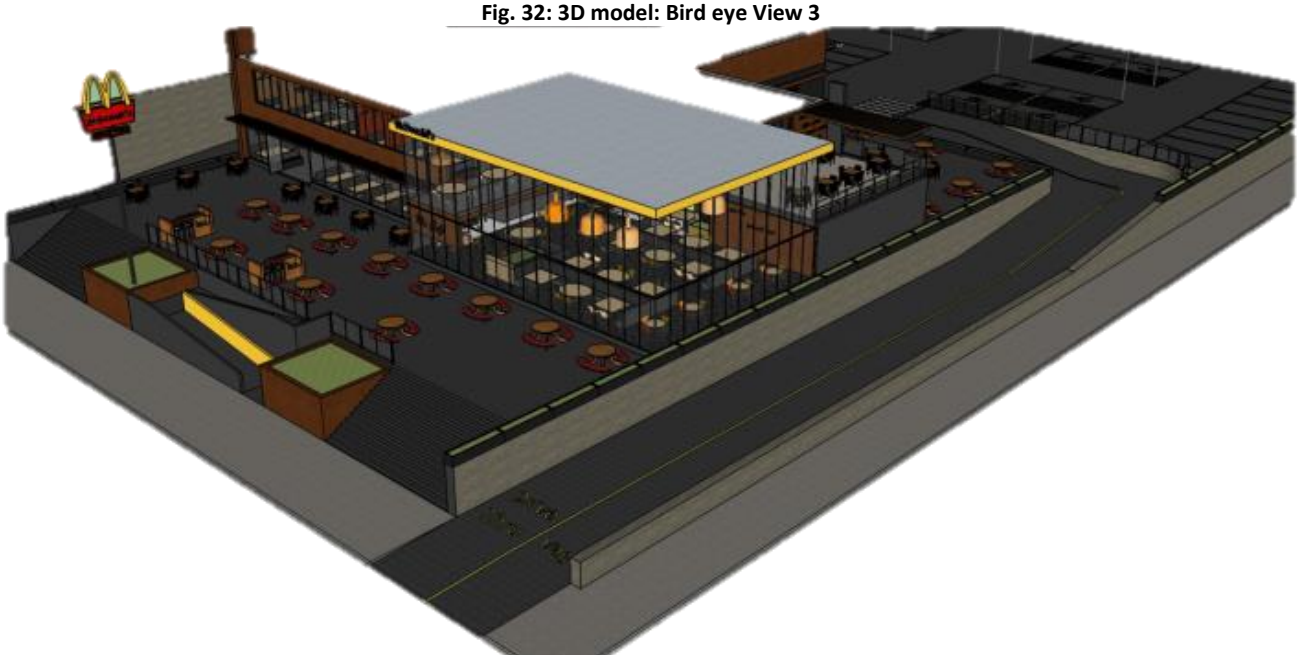


Fig. 33: 3D model: Bird eye View 4

Phase 4: Integrating Game Engine into Architectural Presentation:

Unity3D is a powerful game development tool that has become increasingly popular due to the strengths of its real-time rendering engine, huge feature set and the ability to publish content to Web, mobile and game console platforms.

Google Sketch-Up + Unity 3D is a great combination. Google SketchUp's easy user interface and it's free version and Unity 3d's wide range of flexible tools for designing interactive experiences. But what really matters is the large amount of assets found at Google SketchUp's warehouse at 3dwarehouse.sketchup.com.

4.3.1. The Presentation Process:

After finishing the 2D schematics, it is time to build the 3D model, in order to insert it into the game engine, we will have to export the model from sketchup as *.FBX* file so it can be visualized and then produced to the client.

The sequence made to produce the game:

1- Import the SketchUp model.
2- Import the unity assets into the scene.
3- Scripting.
4- Test the game.
5- Build & run.

Fig. 34: Unity start up screen

4.3.2. Importing the model into the game engine:

Import the 3D model into Unity by dragging the file into the project window.

In the inspector > Model tab Unity supports importing models from most popular 3D applications.

Importing meshes into Unity can be achieved from two main types of files:

1. **Exported 3D file formats**, such as .FBX or .OBJ
2. **Proprietary 3D application files**, such as .Max and .Blend file formats from 3D Studio Max or Blender for example.

4.3.3. Importing Assets:

Model files that are placed in the Assets folder in Unity project are automatically imported and stored as Unity assets.

A model file may contain a 3D model, such as a character, a building, or a piece of furniture. The model is imported as multiple assets. In the Project view the main imported object is a Model Prefab. Usually there are also up to several Mesh objects that are referenced by the Model Prefab.

A model file may also contain animation data which can be used to animate this model or other models. The animation data is imported as one or more Animation Clips.

The Import Settings for a model file will be displayed in the Model tab of the FBX importer inspector when the model is selected. These affect the mesh, it's normal and imported materials. Settings are applied per asset on disk so if you need assets with different settings make (and rename accordingly) a duplicate file.

• Scale - this scale factor is used for compensating difference in units between Unity and 3d modeling tool - it rescales whole file. Normally you can simply set it to 1. Note that Unity's Physics Engine is scaled as 1 unit equals 1 meter. It is important that if you want to have correct physical behavior you should have the model correctly scaled in the original modeling application. If this cannot be done, or you do not have control over the modification of the mesh, the scale of the model can be adjusted here.

• Generate colliders - this will generate a collision mesh to allow our model to collide with other objects.

• Material Naming and Search - this will help automatically setup the materials and locate textures.

Fig. 35: the game character

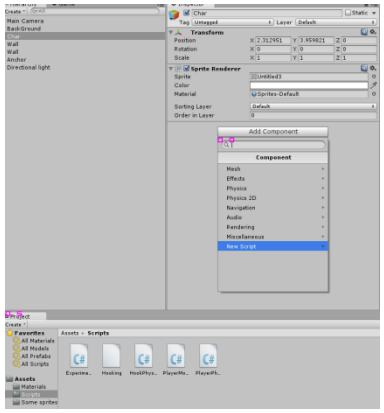


Fig. 36: Script Assigning

Unity also proves to be beneficial for workflows due to its file co-occurrence; once a file, such as an FBX, is loaded into a Unity scene and textures etc. have been assigned/mapped, any change to that FBX file in its root directory will automatically be updated in Unity with the texture maps still intact, hence you can easily update a model in Unity by simply overwriting the FBX with the updated version.

As for the imported assets, we begin with the main character; this character will be the player in the game (the client's avatar). The game is designed to be 3rd person shooter, means that this character will appear during the client's navigation through the building.

Fig. 37: the game car

Also a C# script will be added to this character in order to make it move, rotate, & use different objects (like opening doors or windows).

Game logic based on Open Source .NET platform, Mono: Script with the full strength, speed and flexibility of one of the world's leading programming environments.

Moving, rotating, and scaling objects just takes a single line of code: likewise for duplicating, removing, and changing properties. Everything can be referenced directly, by name or hierarchy, tags, proximity, or touch.

4.3.4. Unity Scripting:

Scripting with Unity brings you fast iteration and execution and the strength and flexibility of a world-leading programming environment. Scripting is uncluttered, straightforward and incredibly fast. In Unity, you write simple behavior scripts in C# or JavaScript (Unity Script). Both languages are easy to use and run on the integrated scripting runtimes.

Fully-integrated script debugging with Mono Develop for both Windows and Mac. Pause your game, do single step line by line, set breakpoints and inspect values.

Unity compiles all scripts to .NET dll files. The .dll files will be jit compiled at runtime.

This allows incredibly fast script execution. It is around 20 times faster than traditional JavaScript and around 50% slower than native C++ code. Unity might take a second to compile all your scripts when you save it. You can see if Unity is still compiling with the small spinning progress icon in the lower right corner of Unity's main window.

```
/// This script moves the character controller forward
/// and sideways based on the arrow keys.
/// It also jumps when pressing space.
/// Make sure to attach a character controller to the same game object.
/// It is recommended that you make only one call to Move or SimpleMove per frame.

var speed : float = 6.0;
var jumpSpeed : float = 8.0;
var gravity : float = 20.0;

private var moveDirection : Vector3 = Vector3.zero;

function Update() {
        var controller : CharacterController = GetComponent.<CharacterController>();
        if (controller.isGrounded) {
                // We are grounded, so recalculate
                // move direction directly from axes
                moveDirection = Vector3(Input.GetAxis("Horizontal"), 0,
                                        Input.GetAxis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= speed;

                if (Input.GetButton ("Jump")) {
                        moveDirection.y = jumpSpeed;
                }
        }

        // Apply gravity
        moveDirection.y -= gravity * Time.deltaTime;

        // Move the controller
        controller.Move(moveDirection * Time.deltaTime);
}
```

Fig. 38: The Script added to the character

Script compilation:

All scripts in "Standard Assets", "Pro Standard Assets" or "Plugins" are compiled first. Scripts in one of these folders can't directly access scripts outside these folders. It is not possible to reference the class or its variables directly, but it is possible to communicate with them using Game Object.

All scripts in "Standard Assets/Editor", "Pro Standard Assets/Editor" or "Plugins/Editor" are compiled next.

These scripts can access scripts from the previous group.

All other scripts outside "Editor" are compiled next.

All scripts that are not in the folders above or in "Editor" are compiled next.

All scripts that are compiled in this step have access to all scripts in the first group ("Standard Assets", "Pro Standard Assets" or "Plugins"). This allows us to let different scripting languages interoperate.

Scripts that are placed in the first group, will take longer to compile, since when they are compiled the third group needs to be recompiled too. Thus to reduce compile times, we put scripts that seldom change into group 1 and scripts that change a lot into group 3.

All scripts in "Editor" are compiled last.

Now we add a camera control script so that the camera can follow our main character in the game.

CameraControl ▸ Awake

```
#pragma strict

var target : Transform;
var distance = -10;
var lift = 20;

function Update () {
transform.position = target.position + Vector3 ( 0, lift, distance);
transform.LookAt (target);

}
```

Fig. 39: Camera Control Script

Testing the game:

For any unexpected errors or failures, a number of tests must be run to insure the functionality of the game; this is a standard procedure done in any game. As expected, a number of errors have accorded, so the team searched on the reason for each error, and applied the solution successfully. Most errors happened in the scripts, so the team had to search on each error by it's code (because each error has it's own code).

4.3.5. Exporting the game: Build & run:

From file menu we choose build & run, or just press Ctrl+B.

Then we select the platform we want to export to (web player).

Finally, a folder will be created with a web page and a unity 3d file, to run the game, we choose the web file.

The file size is about 84 mb, so it can be sent easily to the client as a .rar folder; the client will then have to unpack it in order to operate the game.

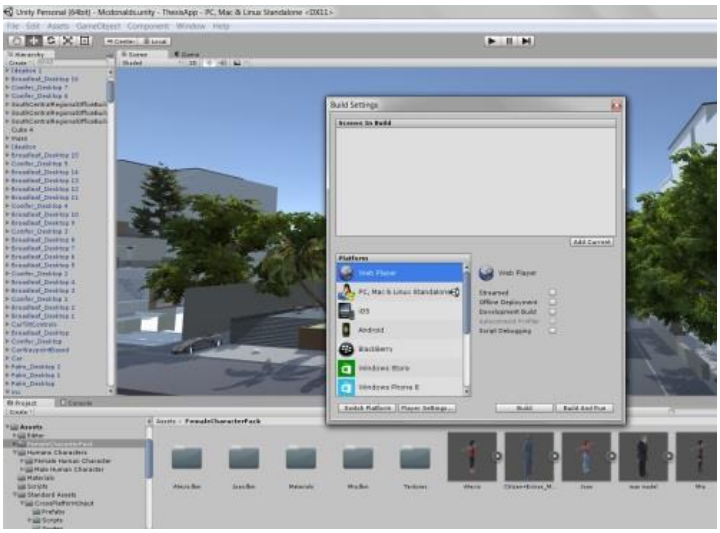


Fig. 40: Exporting the Project

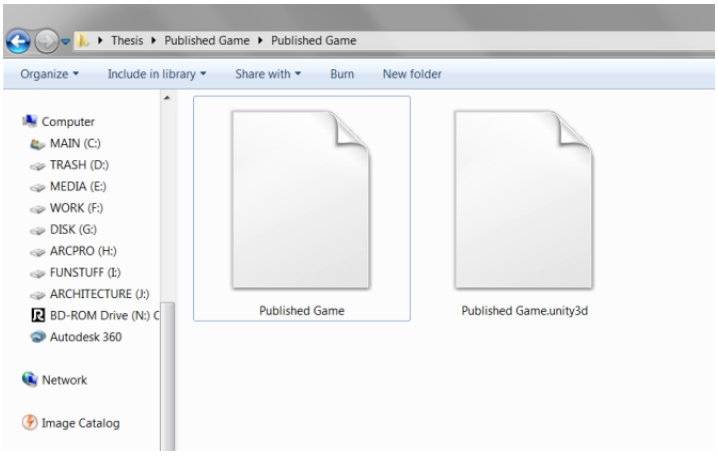


Fig. 41: the Exported Files

And by that, the experiment is finished and now it is time to present the project to the client.

4.4. Phase 5: Publishing the project-The Game:

After exporting the project as a web play, it automatically creates a web page file that will run in most web browsers, this file will be sent to the client in order to open, play and navigate the project to give his feedback on what was done.

When the client receives the web page file (or Link), he will open it for review, a screen will appear to him and he will be asked to install unity web player in order to run the game on the browser.

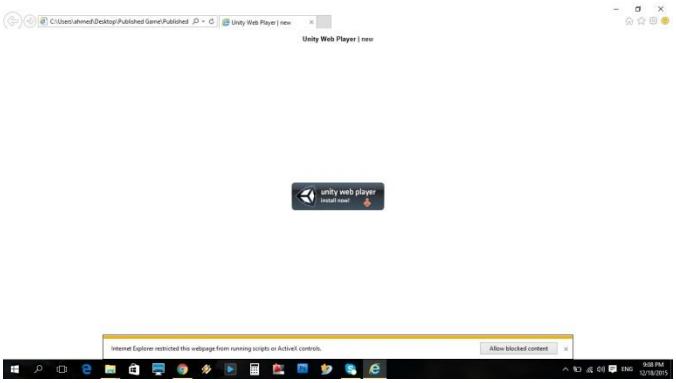


Fig. 42: Unity web player

Note that this is done only once, means that the next time the client will run another game made by unity game engine, and he will not be asked to install the unity web player again.

After installing unity web player, now the game starts and the clients can start playing the game.

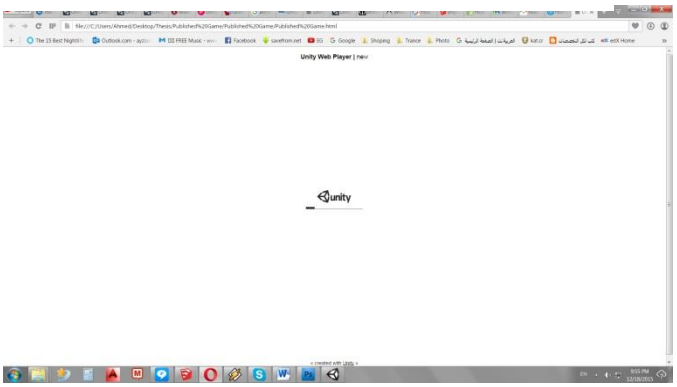


Fig. 43: Installing Unity web player

The game is designed to be played in as following:

1st the client will circulate the project exterior using a dynamic object – a car – to walk through the parking zone & the drive thru lane.

2nd the client will use a character in order to circulate the interior of the project.

Fig. 44: Game Screen Shot 1

As written in the "car motion script", the player (client) will use the arrow keys (up for forward, down for backward, left for turning left, right for turning right) to navigate the outer area of the project.

Then after navigating the drive-thru lane, he will have to park the car in order to begin his character navigation.

Fig. 45: Game Screen Shot 2

Now he will continue the rest of the tour on foot, to do so, the player will use same keys as he used to mobilize the car to move his avatar.

He will exit the parking zone and head towards the restaurant back door walking through the terrace.

Fig. 46: Game Screen Shot 3

Entering the building itself, he will now start his virtual online tour at every corner of the place to insure that it was designed as desired before.

Fig. 47: Game Screen Shot 4

Then he will head to the counter to place an order.

Fig. 48: Game Screen Shot 5

He will receive his order, and head to the seating area inside the building.

Fig. 49: Game Screen Shot 6

The player will change his mind and head for the terrace outside the building

Fig. 50: Game Screen Shot 7

Then, it is time to walkthrough the 2nd floor.

Fig. 51: Game Screen Shot 8

Using the pre-modeled stairs, the player will head to the 2nd floor to investigate it.

Fig. 52: Game Screen Shot 9

Now it Is time for the player to inspect the 2nd floor.

Fig. 53: Game Screen Shot 10

After reaching the 2nd floor, the player will walk towards the 2nd floor terrace.

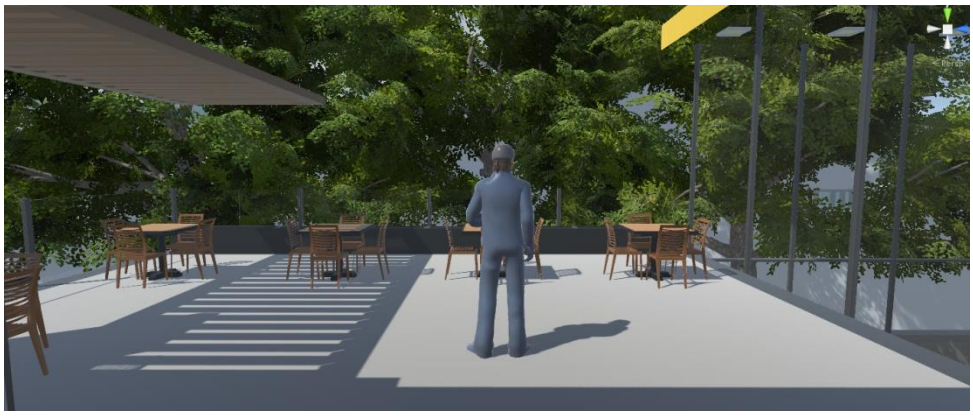


Fig. 54: Game Screen Shot 11

Finally the player reaches the other terrace and by that the tour in the building is over.

Fig. 55: Game Screen Shot 12

The player doesn't need to circle back to the car because he had already seen every corner inside & outside the building, now it is time for the client to give his review of the design to the architects.

5. Study results & analysis

5.1. 1st The I.T. Staff:

They are the first to deal with this software, their objective was simple, to download and install the unity game engine from https://unity3d.com/get-unity/download , on the architects computers.

The whole operation was a success, they encountered no problems, because it is open software, downloading & installing it should be easy for any one.

Then, the I.T. Staff have to download unity assets from the assets store https://www.assetstore.unity3d.com , in order to complete the scene, characters and objects are needed along with environment and landscape elements.

Their feedback on the experiment was positive, no problems or unknown errors were detected at all, from downloading to operating the software.

5.2. 2nd The Project Architects:

Interactive applications allow walking around something that hasn't been built yet or has been lying in ruins for centuries. Even though most Computer Aided-Design (CAD) tools can create 3D visuals, it's hard for clients to really understand what their architect has in mind.

- As for the architects, It is the first time to work with unity game engine, unfortunately; it is not as something they have worked with before, so they had to depend on http://unity3d.com/learn , to try to understand the basics of this software.
- The second challenge was to import & edit the SketchUp model with Unity game engine, so they also had to learn about modeling to edit on the model.
- Importing assets was an easy task, just drag and drop inside the game engine asset's window, and then drag the desired assets into the scene, once they are inside the scene; they had to be scaled to fit the scene.
- Scripting, this was a major challenge, mainly because the working architects lack any experience about scripting and how to deal with java or C#, but in the model needed only 3 scripts (the car-the character-the door),so the team had to download the required scripts from http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html .
- Finally it is time to run the game as a test before exporting it as a web game, this step was important to insure the game functionality, every error emerged, the team had to search on it's cause and the proper solution for it.

5.3. 3rd The Client:

The client was exited to try a game based on his project, using the arrows keys; he kept navigating the project with no help or guidance from the team, it was his first time for him to encounter a demonstration like this, but in the end he was satisfied with the design and the way it was presented by the design team.

The Summary:

This research discussed the ability to use a game engine as a presentation tool, so based on the experiment results; it is now considerable to depend on it in the future projects, the potentials given by this software were encouraging to switch from common used software and presentation methods & tools to the game engine.

Conclusions & Future Perspectives:

- Unity is a great medium for creating a serious game, video game, browser based game.
- The environment is very easy to get used to.
- The support and community are easy to access and use.
- Uses the most recent .Net, PhysX, and graphics frameworks.
- Allows for multiplayer support.
- Can allow for a complete browser-based experience using the lightweight, browser plugin.
- Utilizing the State Synchronization portion of Unity allows the user to update all connected instances of the game so that they all share the same data.
- It is possible to create a persistent world using a server-client interface running Unity.

Recommendations:

1- Trying to emerge the game technology more into architecture use to benefit more from it's capabilities.
2- Add more media (sound effects, animation scenarios) to the game engine platform to bring the project closer to reality.
3- Learn more about programming languages to use it with the different assets in the scene.
4- Depending on more than one game engine like (Unreal – Crytek), to enhance the presentation experience.

Bibliography:

References:

1- Sirbu, D. (2003). Digital Exploration of Inbuilt Architecture: A Non-Photorealistic Approach. *ACADIA 2003*, pp. 235 - 245.

2- Stappers, P. J., Saakes, D., & Adriaanse, J. (2001). On the Narrative Structure of Virtual Reality: An Analysis and Proposed Design. *CAAD Futures 2001*, pp. 125 - 138.

3- Kruijff, E., & Donath, D. (2000). Supporting Shared Architectural Understanding: Spatial knowledge transfer within Virtual Environments. *Proceedings of ACCOLADE 2000 Workshop*.

4- Dorta, Tomás ; Augmented Sketches and Models: The Hybrid Ideation Space as a Cognitive Artifact for Conceptual Design , University of Montreal, Faculty of Environmental Design.

5-Young,R. Michael , O.Riedl,Mark, Branly,Mark , Jhala,Arnav , Martin R.J. , Saretto C.J. ; An architecture for integrating plan-based behavior generation with interactive game environments ; Submitted for Publication in Journal of Game Development.

6- Abdullah Bakarman , Ahmed ; Quality Evaluation Tool for the Design Studio Practice , King Saud University, Riyadh, Saudi Arabia.

7-Lowe,Russell , Newton, Sidney ; Using an Analytics Engine to Understand the Design and Construction of Domestic Buildings ; Faculty of Built Environment, The University of New South Wales, Sydney, 2052,Australia.

8- E. Haque, Mohammed , Dasgupta, Pallab ; Architectural/Engineering Visualization using Game Engine ; Proceedings of the 2008 ASEE Gulf-Southwest Annual Conference , The University of New Mexico – Albuquerque.

9- Conway, Kevin R. ; Game Engines for Architectural Visualization in Design , University of Washington 2011, P9-13.

10- Paul Gee, James ;Video Games, Mind, and Learning , University of Wisconsin-Madison , 2003, P2.

11- Koehler, Tanja, Dieckmann, Andreas ,Russell, Peter ; An Evaluation of Contemporary Game Engines , RWTH Aachen University, Germany , P743-749.

12- Campbell, Dace A. ; A Critique of Virtual Reality in the Architectural Design Process , University of Washington ,Seattle, WA 91895.

13- Miliano ,Vito, Industrielle, Perilith ; Unreality: Application of a 3D Game Engine to Enhance the Design, Visualization and Presentation of Commercial Real Estate.

14- Moloney, Jules , Harvey, Lawrence ; Visualization and 'Auralization' of Architectural Design in a Game Engine Based Collaborative Virtual Environment , University of Auckland and the Spatial Information Architecture Laboratory, RMIT University.

Readings:

1- Soliman Baghdady, Dina ; The Use of Interactive Applications in the Design of Public Buildings , Faculty of Engineering , Ain Shams University.

2- Talaat El Daly , Hazem ; REVISITING ALGORITHMS IN ARCHITECTURAL DESIGN“TOWARDS NEW COMPUTATIONAL METHODS” , Faculty of Engineering , Ain Shams University.

3- Plummer, Jeff ; A FLEXIBLE AND EXPANDABLE ARCHITECTURE FOR COMPUTER GAMES ARIZONA STATE UNIVERSITY, December 2004.

4- W. Knight, Michael , G.P Brown, Andre ; Increasing Design Reflection and Improving Feedback using Wikis , University of Liverpool, UK.

5- Money Harris ,David ,L. Harris ,Sarah ; Digital Design and Computer Architecture , Oxford, UK.

6- Kobayashi, Yoshihiro ; 3D City Model Generator: The Application of Neuro-Fuzzy Systems in CAD, Arizona State University.

7- Kolarevic ,Branko ; Architecture in the Post-Digital Age: Towards Integrative Design , University of Calgary.

8- Corradini ,Andrea , Hanneforth ,Thomas, and Bak ,Adrian ; A Robust Spoken Language Architecture to Control a 2D Game , University of Potsdam ,14415 Potsdam, Germany.

9- Moloney , Jules ; Game Engines and Virtual Design Studios Technology and Pedagogy , The University of Auckland.

10- Orkin , Jeff ; Agent Architecture Considerations for Real-Time Planning in Games , American Association for Artificial Intelligence , 2004.

Glossary:

Game

A game is a formal description of a strategic situation.

Backward induction

Backward induction is a technique to solve a game of perfect information. It first considers the moves that are the last in the game, and determines the best move for the player in each case. Then, taking these as given future actions, it proceeds backwards in time, again determining the best move for the respective player, until the beginning of the game is reached.

Common knowledge

A fact is common knowledge if all players know it, and know that they all know it, and so on. The structure of the game is often assumed to be common knowledge among the players.

Dominating strategy

A strategy dominates another strategy of a player if it always gives a better payoff to that player, regardless of what the other players are doing. It weakly dominates the other strategy if it is always at least as good.

Extensive game

An extensive game (or extensive form game) describes with a tree how a game is played. It depicts the order in which players make moves, and the information each player has at each decision point.

Game theory

Game theory is the formal study of decision-making where several players must make choices that potentially affect the interests of the other players.

Mixed strategy

A mixed strategy is an active randomization, with given probabilities, that determines the player's decision. As a special case, a mixed strategy can be the deterministic choice of one of the given pure strategies.

Nash equilibrium

A Nash equilibrium, also called strategic equilibrium, is a list of strategies, one for each player, which has the property that no player can unilaterally change his strategy and get a better payoff.

Payoff

A payoff is a number, also called utility, that reflects the desirability of an outcome to a player, for whatever reason. When the outcome is random, payoffs are usually weighted with their probabilities. The expected payoff incorporates the player's attitude towards risk.

Perfect information

A game has perfect information when at any point in time only one player makes a move, and knows all the actions that have been made until then.

Player

A player is an agent who makes decisions in a game.

Rationality

A player is said to be rational if he seeks to play in a manner which maximizes his own payoff. It is often assumed that the rationality of all players is common knowledge.

Strategic form

A game in strategic form, also called normal form, is a compact representation of a game in which players simultaneously choose their strategies. The resulting payoffs are presented in a table with a cell for each strategy combination.[12]

Strategy

In a game in strategic form, a strategy is one of the given possible actions of a player. In an extensive game, a strategy is a complete plan of choices, one for each decision point of the player.

Zero-sum game

A game is said to be zero-sum if for any outcome, the sum of the payoffs to all players is zero. In a two-player zero-sum game, one player's gain is the other player's loss, so their interests are diametrically opposed.

[12] Fudenberg, Drew and Tirole, Jean (1991), Game Theory. MIT Press, Cambridge, MA.
Gibbons, Robert (1992), Game Theory for Applied Economists. Princeton University Press, Princeton, NJ.
Myerson, Roger B. (1991), Game Theory: Analysis of Conflict. Harvard University Press,Cambridge, MA.
Nasar, Sylvia (1998), A Beautiful Mind: A Biography of John Forbes Nash, Jr., Winner of the Nobel Prize in Economics, 1994. Simon and Schuster, New York.
Rasmusen, Eric (2001), Games and Information: An Introduction to Game Theory, 3rd ed. Blackwell, Oxford.