Recreating a TransMedia Architectural Location In-Game via Modular Environment Assets

Rui Rodrigues, Nataska Statham, João Jacob, Mikael Fridenfalk

Recreating a TransMedia Architectural Location In-Game via Modular Environment Assets

Nataska Statham[1[0000-0001-9463-1353]], João Jacob[2[0000-0003-2078-3375]], Mikael Fridenfalk[2[0000-0002-3754-172X]], and Rui Rodrigues[2[0000-0003-4883-1375]]

[1] Uppsala University, 621 67 Visby, Sweden, nataska.statham@speldesign.uu.se
[2] University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal, joao.jacob@fe.up.pt
[3] University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal, rui.rodrigues@fe.up.pt
[4] Uppsala University, 621 67 Visby, Sweden, mikael.fridenfalk@speldesign.uu.se

Abstract. Existing architectural locations are often recreated in games using unique "hero" meshes instead of modular assets, which in these cases are commonly perceived as too limited or inaccurate. This applies to real-world locations or, as in this case study, transmedia locations. This study proposes that hero meshes are not always necessary and that modular assets have the potential to recreate even complex architecture. The paper presents a set of development steps for modular assets for game environment art according to a game design lifecycle, and proceeds to demonstrate its potential via a case study. The case study focuses on planning and designing steps; these preliminary results indicate that, when well-designed, modular assets have the potential to recreate complex architectural locations without requiring extensive use of hero meshes. Adopting modular assets instead of hero meshes could potentially reduce the cost and development time of environment art for transmedia games and games featuring real-world architectural locations, as well as increase the reusability of such assets.

Keywords: Transmedia Architecture, Game Environment Art, Modular Assets.

1 Introduction

Modular game assets are often limited to architecture with obvious repetition [1], such as terraced houses, shopping malls, and facades of high-rises. When faced with recreating existing or complex architectural locations in-game, environment artists tend to rely on unique meshes, commonly referred to as hero meshes. Hero meshes, while allowing for a high level of accuracy, are hard to change or customize, have limited or no reusability, and are generally time-consuming to produce [2,3]. For example, in the game Assassin's Creed Unity [4], the cathedral of Notre Dame de Paris took a dedicated environment artist two years to recreate it in-game [5].

Advocates of modularity argue that even complex locations can be recreated in-game using modular assets, as long as they are well-designed [2]. However, there are few published guidelines on how to design and implement modular assets, and game results

often display varying levels of quality. The main problems with poorly designed modular assets are the need for a large number of unplanned extra assets to disguise visual and/or functional flaws as well as excessive visual repetition, which can lead to art fatigue [6].

This paper proposes a set of principles and methodology that support the production of modular assets for 3D games, applied to a case-study. The case-study reproduces a common scenario in transmedia game productions and aims to recreate via modular assets a complex well-known architectural location, namely the police headquarters of Gotham City, part of the Batman universe. More specifically, the version depicted in the TV show Gotham aired between 2014 and 2019 by Fox Broadcasting Company. The goal is to demonstrate that, when well-designed, modular assets have the potential to recreate even highly complex architectural locations in-game.

Section 2 presents the development steps for modular architecture for games as proposed by the authors and according to a game development lifecycle (GDLC). Section 3 introduces the case study as well as available references. Section 4 presents the planning phase and the motivation behind key design decisions. Section 5 presents the preproduction phase, which involves the whitebox and greybox stages, and where the functionality of the assets and the elements of the modular kit are established. The paper concludes with section 6 where the proposed principles and their application in this case study are discussed.

2 Principles of Modular Architecture for Games

There are several steps to design and implement modular assets for a game, involving both environment artists and level designers. A key difference compared to non-modular assets is the need for extensive planning and early testing, where the list of modular assets and their functionality should ideally be established before full production starts as late changes can be time-consuming. In non-modular assets, small changes during production can be easier to implement. Table 1 presents a summary of development steps according to a typical GDLC, proposed by the authors based on observed practices in game development and on established principles of traditional pre-fabricated modular architecture.

Table 1. Development steps for modular architecture for games according to a typical GDLC.

Step	Description
Planning	
Determine the measuring convention	A game should always follow a single measuring convention (metric, imperial, or custom) which should be clearly communicated and available in the game engine [6,7].
Determine the grid size	Game engines rely on Cartesian grids, and modular architecture is built upon multiples of the game's grid size, which is usually determined based on the size of the characters, camera angle, type of gameplay, and the feeling the game is trying to evoke [2,6].

Step	Description
Establish the naming convention	A clear and well-documented naming convention is fundamental for modular architecture for games. As names are likely to get long, using abbreviations or omitting vowels are standard.
Define the types of modular kits	Large-scale games may require several kits to organize a variety of architectural styles, construction techniques, and factions. As a rule of thumb, fewer kits with more components tend to be more cost-effective than more kits with fewer components [2].
Define the types of modularity	The types of modularity influence how the assets are designed; most games rely on sectional, component swapping, and component sharing modularity [6].
Pre-production	
Whitebox	As with traditional level design, pre-production starts with whiteboxing, which for modular architecture should be approached as a vertical slice. The goal is to determine as much as possible all types of levels needed, rather than designing the actual levels. These will guide the planning and design of the modular kits [2,6].
Refine types of kits and types of modularity	If needed, the types of modular kits and types of modularity may be refined. It is common for two or more modular kits to be consolidated at this stage [2].
Plan the modular kits:	
a) assets per modular kits	Create a detailed list of assets per modular kit. While variations in style and decorative details may be added later, all the types of basic assets should be listed.
b) tiling direction	Sectional modularity, where assets are tiled side by side, is the most common type of modularity in games [6]. It is important to determine the tiling direction of each asset: horizontal, vertical, or both.
c) level of granularity of the kit	Modular kits that rely on smaller individual meshes tend to be more flexible and offer greater visual variation. On the other hand, when assembling the levels within the engine, too many small assets generally requires a longer assembly time and can result in more errors [2].
d) use of prefabs or not	Kits with small granularity may benefit from the intermediate step of pre-assembly as larger prefab components in engine prior to level design to facilitate the level assembly [2].
e) position of pivot points	All 3D meshes have a pivot point from which they can be moved, rotated or scaled. It is important that all pivot points within a game follow a common logic to facilitate tiling [6,8].
f) standard size of transitions	Different kits can be connected seamlessly by adopting standard sizes for transitions such as doorframes and stairs [2,9].
g) transitions as plug and socket or special meshes	Transitions are further optimized by using a plug and socket system, where the trims of the transitions are separate meshes that plug into the sockets of door and window frames [2,9].
h) components shared between kits	Modular kits rely on additional components for variation and to minimize art fatigue. These include modular pipes, wires and structural frames, decals and greebles, and set dressing props. These components are often shared between kits [3].

Step	Description
Greybox	Once the modular kits have been planned, environment artists create placeholder grey meshes (untextured) with the correct dimensions and layout to test the functionality of the kits [2]. These replace the whitebox meshes, allowing quick in-game testing and iteration.
Early Production	
Basic/vanilla variation of the kit	The focus of early production is to confirm the design of the kits through a first and basic/vanilla version of each kit. Because there is only one variation of each kit at this point, it is easier to implement changes or to add elements that might have been overlooked during pre-production [2].
Functional polish	Early production should finish with a functional polish of the modular kits, where the basic variation of each kit is in engine replacing the placeholder versions. The basic variation must be deemed functionally sound before further variations are added [2].
Production	
Create and implement variations	This stage focuses on adding variations based on the basic/vanilla version, usually combining mesh and texture variations [3].
Create and implement set dressing and props	Set dressing components and props are also added. These might be modular or stand-alone assets [2,3].
Polish	
Visual polish	Previously identified visual weaknesses should be addressed. New meshes and textures should generally not be added at this stage [2].
Optimization	Standard game graphic optimizations such as packing texture maps, adjusting LODs, and graphic quality balancing [2].
Correcting bugs	Previously identified bugs should be corrected.

3 Case Study: Gotham PD

The case study focused on the design and functional integrity of the modular assets, and therefore on the phases of planning and pre-production. It was developed using Unreal Engine 4 release 4.24 and Autodesk 3ds Max 2020. The goal was to reproduce in-game via modular assets an existing architectural location that would ordinarily be created via hero assets. Mimicking a game development environment, the authors relied on pre-existing blueprints and footage from the TV show as references. As it is unusual for game developers to have access to TV or movie sets, the lack of first-hand references placed the authors in similar circumstances.

4 Planning

Game developers relying on modular assets for environment art instead of hero meshes should be prepared to invest more time in planning, as layout and design must be decided early, before production and asset creation has started. There is generally little leeway for changes in late development, and late changes to the modular assets can be expensive. On the other hand, changes to the layout and level design are generally easy due to the flexibility of modular assets, compared to that of assets created as a single hero mesh [2]. For the case study, since Batman is an on-going game franchise, the authors relied on characteristics of the already released games to guide the decision-making process. Of particular significance, Batman are action-adventure games with a core mechanic of jumps, moving and combat between different heights.

Planning included quick sketches of key angles of the location based on reference images and blueprints [1], taking into account the ceiling height needed for the core mechanics. The grid size was set at 1.5 times the height of the character; larger grid sizes can be beneficial for games that require broad character movements and jumps and is in accordance with previous game instalments of the Batman series. The average male game character is 1.8 meters tall, usually rounded at 2 meters to calculate environment sizes. At 1.5 times the character height, the minimum grid size was set at 3 meters. This allowed for grid size variations of 3, 6, 9, and 12 meters, with smaller structures and props of 0.5, 1, and 1.5 meters.

During planning, the following were decided: the measuring convention, the grid size and allowed variations, the naming convention, the types of kits, and the types of modularity. These are described in Table 2.

Table 2. Description of the main characteristics of the modular assets for the case study.

Characteristic	Case study standard
Measuring convention	Metric
Grid size	3 meters (1.5x the height of the main character)
Grid size variations	3, 6, 9, 12 meters, and smaller structures of 0.5, 1, and 1.5 meters
Naming convention	NameOfKit_TypeOfAsset_Dimension_Characteristic_Variation.
Modular kits	The police station is planned as part of a larger modular kit for public buildings, abbreviated to Pub for the naming convention
Types of modularity	Sectional modularity with vertical and/or horizontal tiling, mix modularity for pillars

5 Pre-Production

Based on the combination of blueprint, sketches, and reference images, pre-production started with a rough layout of the location within Unreal Engine using brushes [1], a technique commonly referred to as whiteboxing [2,6]. The initial step was to adapt the original measurements into the grid size convention of 3, 6, 9, and 12 meters. This proved straightforward, with few modifications required.

The differences between the original blueprint and the whitebox included:

- simplified small irregular corners to facilitate character navigation,
- expanded the depth of enclosed rooms that were originally too small for a game and a potential source of bugs in the form of characters getting stuck to each other or to the surrounding geometry,
- stairs sizes were standardized,

The level was play tested for navigation, movement, and possibilities of jumping and combat sequences. During this stage some of the furniture was rearranged to facilitate character navigation and the arcs of the mezzanine were removed to avoid blocking the character's path when jumping and performing ranged attacks. Based on the whitebox level, further decisions were taken regarding the design of the modular kit as well as a list of assets and their description. These steps are described in Table 3.

Table 3. Description of the main characteristics of the modular assets for the case study.

Characteristic	Case study standard
Level of granularity	High level of granularity using prefab groups.
List of assets	Where possible, assets were combined to minimize repetition.
Pivot points	Place at the back right corner; for radial meshes in the center of the circumference.
Transitions	The dimensions of windows and stairs were standardized.

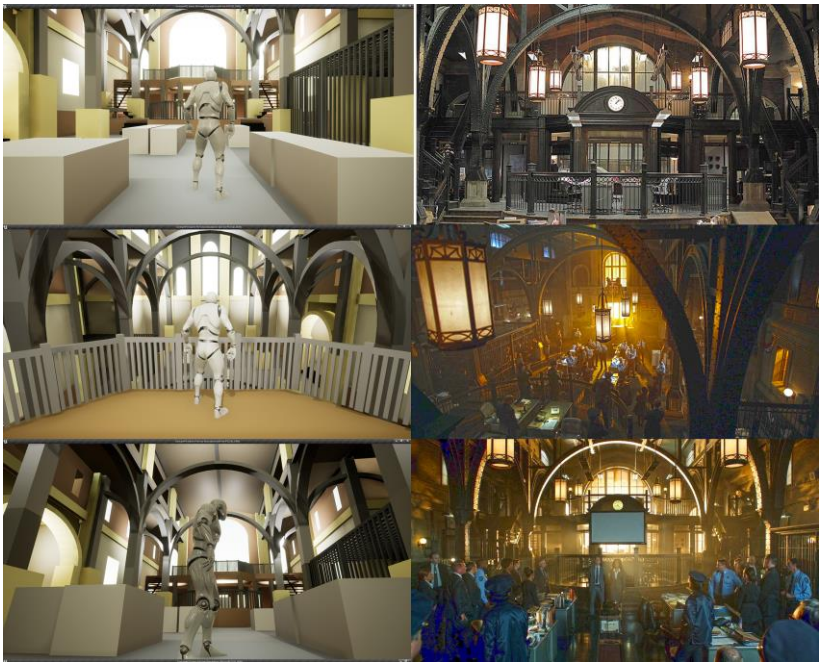

Fig. 1. Left: Greybox level in Unreal. Right: Reference images from the TV show.

5.1 Greybox

Once the list of assets was finalized, the whitebox geometry was replaced with greybox meshes. Greybox meshes are simplified placeholder versions of the final meshes with the same size and layout but without a high artistic investment and are typically created in a 3D modelling software, in this case Autodesk 3ds Max 2020. They allow environment artists and level designers to quickly test the functionality of modular assets, including the position of pivot points, their dimensions, and their modular tileability. Fig. 1 compares the greybox level with references of the TV show.

6 Conclusion and Future Work

When faced with complex or well-known architectural locations, game environment artists often choose to recreate them using hero meshes instead of modular assets. While hero assets provide greater freedom and higher potential for accuracy, they tend to be time-consuming to produce and have limited reusability [1,2]. Changes in layout can be expensive to implement on finished hero meshes. Modular assets on the other hand have high layout flexibility, greater reusability, and tend to be faster to produce [6]. This case study aimed to test whether it was possible to recreate in-game a well-known transmedia architectural location using modular assets instead of hero meshes.

Based on a set of production steps applied to the different stages of a game development lifecycle, this case study demonstrated that, when well-designed, modular assets have the potential to accurately recreate in-game complex existing architectural locations. The case study focused on the design and functionality of the modular kit, corresponding to the planning and pre-production stages. It highlighted the flexibility of modular assets and the importance of early testing and integration between environment art and level design. By testing the whitebox and greybox versions in-game, it was possible to quickly adapt the layout to facilitate the gameplay, for example by moving objects that might block the characters' movement and increasing the footprint of specific areas. Whereas these adaptations were important for gameplay, they were also minimal and localized. The corresponding original TV show and in-game versions of the location match closely and are easily recognizable. Modular architecture was flexible enough to recreate even complex architectural features, such as the structural arcs and elaborate windows.

By recreating the level using modular assets instead of hero meshes, the complexity of each mesh was reduced. Each modular mesh is relatively simple, and the modular kit has the advantage of flexibility and reusability. For example, if level designers wish to change or expand the police station, modular assets enable such changes at a low production cost, whereas a hero mesh would require a dedicated team of environment artists to create the new or modified location from scratch.

This case study reinforced the importance of planning and early testing when designing modular assets for games. It is important that testing takes place in the game engine, and that there is clear documentation of the measurements, standards, and guidelines for the modular kit as well as a clear and detailed list of assets. Time invested in early testing can help to identify mistakes or weaknesses in the modular assets that

could easily be corrected during pre-production, but that might be costly and time-consuming to change at later production stages.

The authors believe this case study indicates that the proposed set of development steps has the potential of being a helpful guideline for game developers and researchers interested in modular assets for games. The steps were flexible enough to be easily adapted to the specific needs of the case study, while detailed enough to ensure that all key decisions were made at the appropriate time. Further testing in a full game development production is recommended. To better evaluate its impact in production, ideally the same or similar game locations should be developed by experts using a modular and a non-modular approach, with results compared in terms of production time, impact in game performance, and visual quality. The principles should also be regularly revised to account for changes in production pipelines, as might be expected from the upcoming release of the next generation of consoles and engines, such as the upcoming Unreal Engine 5 or from the introduction of new technology, such as wider-spread use of photogrammetry-based meshes in games.

References

1. Suanto, W., Martyastiadi, Y. S.: Modular Technique of 3D Modeling and Procedural Texturing for 3D Game Environment Design of "Jurnal Pahlawan". In: International Conference Intermedia Arts Creative Technology, pp. 5–12. https://doi.org/10.5220/0008525200050012. Yogyakarta, Indonesia (2020)
2. Burgess, J., Purkeypile, N.: "Fallout 4's" Modular Level Design. In: Game Developers Conference GDC 2016. InformaTech, San Francisco, CA, USA (2016).
3. Burgess, J., Sergeev, A.: Building Huge Open Worlds: Modularity, Kits & Art Fatigue. In: 80 Lev. (2018). https://80.lv/articles/building-huge-open-worlds-modularity-kits-art-fatigue, last accessed 2021/01/25.
4. Ubisoft: Assassin's Creed Unity [Video Game] (2014).
5. Makedonski, B.: One dev spent two years making the Notre Dame in Assassin's Creed Unity. In: Destructoid. (2019). https://www.destructoid.com/one-dev-spent-two-years-making-the-notre-dame-in-assassin-s-creed-unity-282133.phtml, last accessed 2020/11/13.
6. Perry, L.: Modular Level and Component Design. In: Game Dev, pp. 30-35. (2002)
7. Burgess, J., Purkeypile, N.: Skyrim's Modular Approach to Level Design. In: Game Developers Conference GDC 2013. InformaTech, San Francisco, CA, USA (2013).
8. Mader, P.: Creating Modular Game Art For Fast Level Design. In: Gamasutra. (2005). https://www.gamasutra.com/view/feature/2475/creating_modular_game_art_for_fast_.php. Last accessed 2011, last accessed 2021/03/02.
9. Smith, R.: Prefab architecture : a guide to modular design and construction. John Wiley & Sons, Hoboken, New Jersey (2010).