Entertainment Computing 41 (2022) 100476

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom

Game environment art with modular architecture

Wilhelmina Zoe Statham [a,b,*], João Jacob [b], Mikael Fridenfalk [a]

[a] Uppsala University, 621 67 Visby, Sweden
[b] University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

ARTICLE INFO

Keywords:
Art Fatigue
Game environment art
Game development lifecycle
Level design
Modular architecture
Modular assets
Modular kits

ABSTRACT

3D games routinely turn to modular architecture as an optimization technique for environment art and level design, although there is little documentation of how to implement it. As a result, many games struggle to apply it effectively, contributing to problems such as art fatigue and production delays. This investigation proposes a set of unified principles of modular architecture for 3D game environment art based on traditional architecture and game examples. It finds that the use of a uniform grid and standard measurements are key as well as extensive planning, and that assets organized as modular kits can be more cost-effective and facilitate flexible level design. It concludes with a series of steps per development phase of a game's lifecycle aimed at game environment art, with emphasis on early planning and testing, and suggests that close collaboration between environment artists and level designers is key to develop effective modular assets.

1. Introduction

Although modular architecture is a recurrent video game optimisation technique, it is poorly documented outside of a handful of developers' postmortems such as Bethesda's panels at the Game Developers Conference (GDC). These are not always transferable to other projects, as reflected by the varying level of quality of modular architecture in games. When poorly designed, it fails to achieve the intended level of optimization and often results in excessive visual repetition and art fatigue[1], which can contribute to spatial disorientation [1]. However, when well designed, modular architecture has the potential to speed production and effectively populate large game maps without requiring extensive teams of environment artists. Art fatigue can be minimized and repetition can be used advantageously by skilled level designers [2].

This paper aims to present how 3D game environment art with modular architecture can be advantageous and proposes a unified set of principles for developers. To do so, the authors first relied on a closely related field that is extensively documented: that of traditional modular architecture and its counterpart of pre-fabricated architecture. Next, focus turned to how modular architecture is commonly implemented in game environment art. The published documentation here is scarce and mostly in the form of panels at trade conferences, trade magazine articles or informal tips by industry experts. To complement this limited body of publication, a representative selection of 3D game examples were also directly analysed (Table 1). These were selected based on their high content of modular architecture and correspond to (1) 1st and 3rd person open world games, (2) 1st and 3rd person linear games, with a focus on space exploration, (3) city-building games.

Open world games are commonly 1st or 3rd person, and sometimes offer both options. They include games within the genres of role-playing games (RPG), action and adventure. Typically these games feature hundreds of hours of gameplay supported by vast maps with a profusion of interactive locations that players can explore seemingly unguided, without necessarily being connected to a mission or game objective [3,4]. While it is common for parts of the map to unlock as the player progresses with the main story, players are seldom restricted to or required to navigate through specific paths only. Gimmicks such as loots, collectable items, points of interest, and interactions with NPCs encourage players to explore the game world outside of mission objectives [5,6]. Typically open world games feature a mix of natural and urban environments.

Unlike open world games, linear games prompt players to follow a specific path from which they are discouraged to deviate. They

* Corresponding author at: Uppsala University, 621 67 Visby, Sweden.
E-mail addresses: nataska.statham@speldesign.uu.se (W.Z. Statham), joao.jacob@fe.up.pt (J. Jacob), mikael.fridenfalk@speldesign.uu.se (M. Fridenfalk).

[1] Art fatigue is not unique to games, and can be experienced when a person is overexposed to repetitive or similar visual stimuli: art exhibitions, museums, large shopping complexes, and even zoos battle this phenomenon. It is triggered by mental fatigue and characterized by a decrease of interest, boredom, information overload, looking for shortcuts, limited attention span, and poor decision-making capacity, eventually leading to the abandonment of the activity [12].

https://doi.org/10.1016/j.entcom.2021.100476
Received 6 October 2020; Received in revised form 17 December 2021; Accepted 17 December 2021
Available online 27 December 2021

Table 1
List of analyzed games. When a version is not specified, the focus was on the most recent instalment of the franchise as of the writing of this paper.

1st and 3rd open world	1st and 3rd linear	City building
Elder Scrolls	Mass Effect	Banished
Fallout	Elite	Planet Zoo
Grand Theft Auto (GTA)	Dead Space	Frostpunk
Saints Row	X universe	Cities: Skylines
Tom Clancy's The Division 1	EVE Online	Surviving Mars
Assassin's Creed		Planet Base
		Tropico
		Anno
		Civilization

Table 2
Comparison of modular characteristics between traditional architecture, open world games, linear games, and city building games.

Characteristic	Traditional architecture	1st/3rd open world	1st/ 3rd linear	City building
Single measuring system	–	✔	✔	✔
Minimum (grid) size	✔	✔	✔	✔
Preference for regular grids	✔	✔	✔	✔
Can follow irregular grids	–	✔	✔	–
Modularity drives the design	✔	–	–	✔
Design for reusability	✔	✔	✔	✔
Extensive planning	✔	✔	✔	✔
Late changes are expensive	✔	✔	✔	✔
Types of modular assets:				
☐ components	✔	✔	✔	-
☐ non-volumetric panels	✔	✔	✔	-
☐ volumetric modules	✔	✔	✔	✔
Favours fewer parts/ higher level of pre-assembly	✔	–	–	✔
Type of modularity:				
☐ component-sharing	✔	✔	✔	✔
☐ component-swapping	✔	✔	✔	✔
☐ cut-to-fit	✔	-	-	-
☐ mix modularity	✔	✔	-	-
☐ bus modularity	✔	-	✔	✔
☐ sectional modularity	✔	✔	✔	✔
Standard connections/ transitions	✔	✔	✔	✔
(Most/all) modules connect to a standard transition	–	–	–	✔
Joints or trims between modules	✔	✔	✔	–
Customization through variation of parts/ meshes	✔	✔	✔	✔
Customization through variation of colours/ materials	✔	✔	✔	✔
Modular functional components add variation	–	✔	✔	–
Modular decorative components add variation	✔	✔	✔	✔
Light sources typically embed in modules	–	–	✔	✔
Easy to customize after assembly	–	✔	✔	–

traditionally feature close-quarter combat through a series of intermittent narrow passages or corridors. Common linear genres are action, adventure and sometimes puzzle games [3,7]. In terms of settings, space exploration is a recurring setting, and one that lends itself well to modularity [8].

City building games are designed around the principle of modularity, giving the player the freedom to create their own game worlds. In game design studies, they are referred to as construction and management simulations (CMS), and challenge players to construct, build or manage within a set of economic and time constraints [3]. The player is provided with a set of available buildings, usually organized by functionality. As the player progresses through the game, the range buildings expands as more advanced resources become available and/ or as the player advances to a different time period [9].

Section 2 summarizes findings from traditional architecture and from the games described above. Section 3 in turn discusses problems such as production delay and art fatigue, followed by section 4, where the authors present their proposed principles of modular architecture for games, and section 5 that contextualises the proposed principles within a game development lifecycle and demonstrates their application via a quick prototype. Section 6 concludes with recommendations and suggestions for automation and specialized tools.

2. State of the art

Modularity is an approach that in traditional architecture facilitates the off-site construction of large building sites at comparatively lower costs and faster turnaround than non-modular onsite construction. It can be scaled to individual buildings such as residential single-family units through the use of prefabricated modular kits that are often customizable through pre-designed variations [10,11]. The first use of modular architecture dates back to the English colonization and over the centuries modular architecture has been associated with the need for fast and cheap construction. This gave it a negative association with low-quality construction methods and buildings that do not always consider the needs of the individuals and of the environment. In large construction sites, modularity can lead to excessive repetition, which in games is referred to as art fatigue [12]. Modern architecture and the Japanese tradition of modular housing endeavour to change this perception, with modern modular architecture striving for sustainability and mass-customization [13,14].

More than an alternative approach to construction, modularity encourages a different design logic and encourages:

- Systems that are composed of separate components (modules) that can be connected or integrated together.
- Systems that allow components to be added or replaced without affecting the rest of the system.
- Systems that can create spaces of differing scale through repetition of components.
- A "modular architecture" easily allows the addition or subtraction of components and can enhance the flexibility of usage and maintenance of a built structure [15].

To enable modularity, traditional architecture relies on a series of principles that are transferable to modular architecture for games. It primarily relies on a uniform Cartesian grid and a basic unit or building module from which other measurements are derived to ensure fitting and modular continuity. Inherent to its success is the standardisation of fittings, materials, and components, without which compatibility between modular elements can be compromised [13]. When designing modular traditional architecture, it is important that modularity is part of the design aesthetic, and not an after-thought. It requires early planning and decisions are generally locked early as late changes tend to be very expensive and hard to implement. Modularity can be realised at different levels of pre-assembly, from individual components, to pre-assembled non-volumetric (2D) panels or sub-assemblies, to volumetric (3D) modules, or as pre-assembled whole building units. The level of granularity affects the number of parts per build and also the

Table 3
Characteristics of modular architecture per type of game analyzed.

Characteristic	1st/ 3rd open world	1st/ 3rd linear	City building
Grid size	~2x character size	~1.5x character size	1 building unit
Types of kit	By type of construction and by faction	By type of space and by faction	By functionality and by era
Level of granularity	Mix of modules, panels and components. Pre-assembly of modules in engine	Mix of modules and panels	Each building is a module
Structural frames	Individual meshes, doubles as trims	Integrated within the modules	N/A
Direction of modularity	Horizontal modules, horizontal and vertical panels, modular pipes and utilities	Horizontal modules, horizontal and vertical panels.	Continuous horizontal and vertical modularity, intermittent horizontal modularity
Types of modularity	Sectional, component sharing, component swapping, mix modularity	Sectional, component sharing, component swapping, bus modularity	Sectional, component sharing, component swapping, bus modularity
Transitions	Standardized doorways and windows with plug and socket system	Mostly standardized doorways	Standardized connection to roads
Variations	Mesh and texture variations, decals, greeles	Mesh and texture variations, decals, greeles	Sectional modularity within the module, mesh and texture variations
Damage	Mesh variation, decals, texture	Mesh variation, decals, texture	Mesh variation, texture
Light sources	External	Mostly embedded, can also be external	Embedded

types of modularity that are possible. Generally in traditional architecture, a higher level of pre-assembly (lower granularity) via volumetric modules results in less parts and faster assembly on site, but there are considerations with transport and logistics which do not translate into game development [10,15,16].

When reviewing the use of modular architecture for game environments, these principles were also observed in environment art and level design, with some adaptations depending on the game. While some characteristics such as basic building unit and favouring Cartesian grids are universal to all modular constructions [2,13], others tend to be more case specific. In particular, the authors noticed different levels of pre-assembly and granularity between 1st and 3rd person open world games, 1st and 3rd person linear games, and city-building games, where open world tend to favour high granularity non-volumetric and component meshes, linear games tend to incorporate a higher level of pre-assembly and volumetric meshes, and city-building games commonly rely on volumetric and whole buildings as a single meshes.

Modular game environment art incorporate elements geared towards preventing art fatigue, such as texture variation, decals and greebles, as well as lighting and set-dressing via props. Functionality is very important when designing modular architecture for games, and all meshes must respect their assigned grid space, follow the same pivot point logic, and standard transitions. To organize the several architectural meshes needed to develop a full game, the use of modular kits with consistent design logic is recommended [2,17]. Table 2 provides a comparison between the modular characteristics of traditional architecture and the types of games reviewed.

Examining games specifically, these characteristics can be further detailed (Table 3).

Although at first glance 1st and 3rd person open world and linear games approach modular architecture similarly, the implementation and planning differ. In linear games, the modules tend to be more pre-built and include both functional and decorative elements embedded within the surface of the module [17]. This arguably facilitates the style of gameplay of linear games, which generally require the player to move at a faster pace and not be distracted by random things to explore. Open world games on the other hand tend to encourage the player to fully explore the game world [3,6]. Functional and decorative elements are normally externalized from the modules as props or overlapping modular elements, creating more points of interest. Often these elements also contain collectible or inventory items.

When designing modular kits for open world games, developers are generally trying to fill large maps and sometimes recreate real-world locations [17]. This requires flexible kits that can accommodate a large array of types of spaces and may benefit from being organized by the type of construction instead of being organized by the type of space. In urban centres, for example, a single robust kit can be used to create shops, offices, schools, post offices, and other small public buildings that share the same construction style, being more cost-effective than creating a single kit for each type of location [2].

Modular kits for linear games on the other hand tend to contain fewer types of spaces and there is generally less onus to recreate real-world locations. Usually a linear game features a smaller range of types of spaces, which are designed to emphasize specific gameplay strategies. These spaces are often repeated throughout the game several times with increased levels of difficulty [17]. Thus a linear game usually requires fewer types of modular kits, and the kits tend to be organized by type of space. For both linear and open world games, different factions, races, or time periods might require either new kits or variations within the kits.

City building games on the other hand tend to design the whole complex of buildings as a single kit, or to break them down by eras or civilizations. There are two levels of modularity in city building games: the modularity between buildings, where side by side buildings must form continuous city blocks; and modularity within buildings, where variations can be achieved through component swapping. While the developer has full control of the design of each individual building, it is the player who decides where to place the buildings and which buildings neighbour each other. In essence the player assumes the role of level designer [9], and city building kits must be carefully planned and tend to incorporate pre-established rules to prevent undesirable outcomes.

As with traditional modular architecture, modular architecture for games relies on careful and advanced planning. The approach towards level design itself is different with modular assets, and while level designers may start with standard whiteboxing for early planning, these are developed into levels using the range of pre-existing modular assets, rather than by creating custom hero meshes [17,18]. Late changes to the modular kits are both hard and expensive to implement, and it is generally recommended that the design of the kits and types of assets within the kits involve both environment artists and level designers. An effective modular kit must work both aesthetically and functionally, supporting both the visual style and the gameplay [2,19].

3. Common issues with modular architecture in games

Functional problems with modular architecture in games may derive from lack of planning, insufficient early testing, or poor communication. Problematic kits are often not flexible enough, or else they lack gameplay features such as covers or adequate access points [2,19]. These can be remedied by expanding the kit, adding hero assets, or by limiting the gameplay. The latter reduces the scope of the game, whereas the former adds extra work, causing possible production delays.

Aesthetic problems can arise from a lack of visual continuity, forming a functional structure that is nonetheless visually incoherent. This is often due to lack of experience, poor architectural knowledge, and

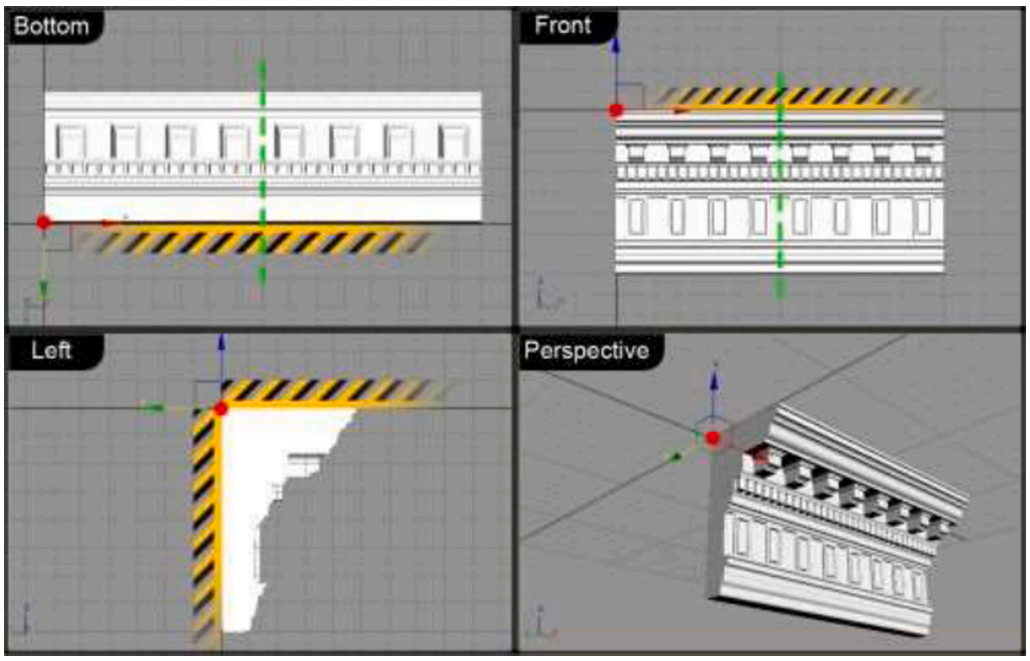


Fig. 1. All tileable modular meshes should have consistent pivot point placement at the corner, as in this example of a tileable crown molding (P. Mader, Creating Modular Game Art For Fast Level Design, Gamasutra. (2005). https://www.gamasutra.com/view/feature/2475/creating_modular_game_art_for_fast_.php. Last accessed 2011 (accessed March 2, 2020).).

insufficient testing. These can also be remedied by adding new assets to the kit, often of a "make up" nature, such as extra trims and decorative elements [20]. This also calls for extra work and can lead to production delays.

While the former are linked to lack of experience or poor implementation, another problem afflicts modular environments even with experienced teams: repetition. Modular environments optimize the game through repetition of assets, and if this repetition is not carefully balanced by enough asset variation, distinctive layouts, and adequate set dressing, it can become noticeable [1]. When the visual repetition is so noticeable that players are distracted by it, it can lead to art fatigue. Artists experienced with modular assets do not see this inherent repetition as necessarily negative. Repetition is expected in traditional architecture. It can add rhythm and desirable repetition (repetition as a principle of design) to a location [2,13]. To break the excessive repetition, artists rely on set dressing, decals, lighting, mesh, and texture variations. Individual assets should be placed with different rotations, mirrored, and varying the grouping and proximity of assets [2,17].

4. Proposed principles of modular architecture for game environment art

The following are proposed by the authors as unified principles of modular architecture for game environment art, offering a set of guidelines that are comprehensive while still flexible enough to be adapted to each game project. They reflect standard practices in game development (such as descriptive naming conventions), observed characteristics of modular architecture for game environments, and transferable practices from traditional architecture.

4.1. 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. 3D modellers and animators are advised to set the measuring convention as the default of their 3D software [17,19]. While this might seem an obvious procedure, it can be complicated by the international character of game development teams, where some team members are likely to be uncomfortable with the measuring convention adopted. More so in large productions involving multiple studios and outsourcing partners spread over several countries.

4.2. Grid

Adhering to a regular and uniform grid lies at the heart of modular assets. Even games that seemingly feature a flexible modular structure are built upon the principle of a regular grid. While games are constantly finding new ways to visually disguise the grid, within today's technology following the same regular grid for the entire game project is still essential for modularity [13,19]. To break rigid angles, kits can plan trims to fill the break in continuity. Another technique is local grid rotation within the engine, so that different segments of the level can follow different grid angles [2,17].

4.3. Grid size

The grid size should correspond to the minimum build unit. It depends on the camera angle, type of gameplay, and the feeling the game is trying to evoke [20]. Platforming games might benefit from a larger grid size to accommodate the player's freedom of movement, whereas packed 1st person shooters and horror games benefit from the sense of confinement of a smaller grid size [17]. For city building games, the grid size corresponds to the smallest possible building and is usually also the width of a road.

All modular assets should be calculated as multiples of the grid size. For example, if the minimum grid size is set at two meters, modular architectural assets may be four, eight, twelve, or sixteen meters, as these will tile continuously. But an asset of three meters would break this pattern. Fractions of the grid size following the same logic may be used to calculate the dimensions of modular furniture, props, and trims [2]. A well-established grid size is important not only for environment art and level design but also for animators and programmers. As characters must interact with these environments and props, standardized dimensions can optimize the game development [2,19].

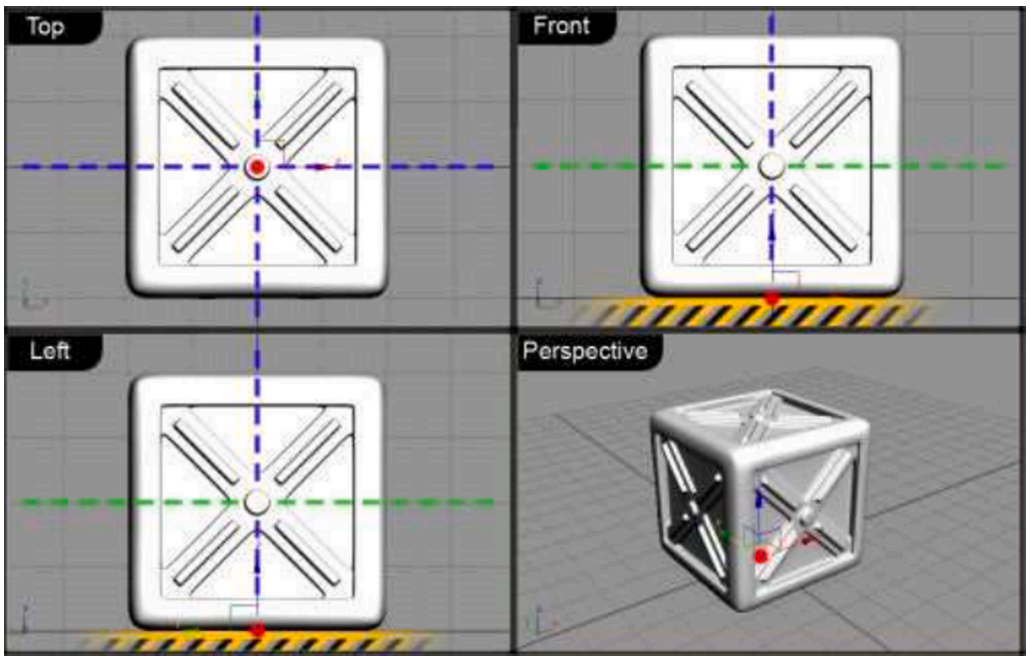


Fig. 2. Modular meshes and props that can rotated around their center axis tend to have the pivot point at their bottom centre, as in the example of this prop (P. Madder (2005).).

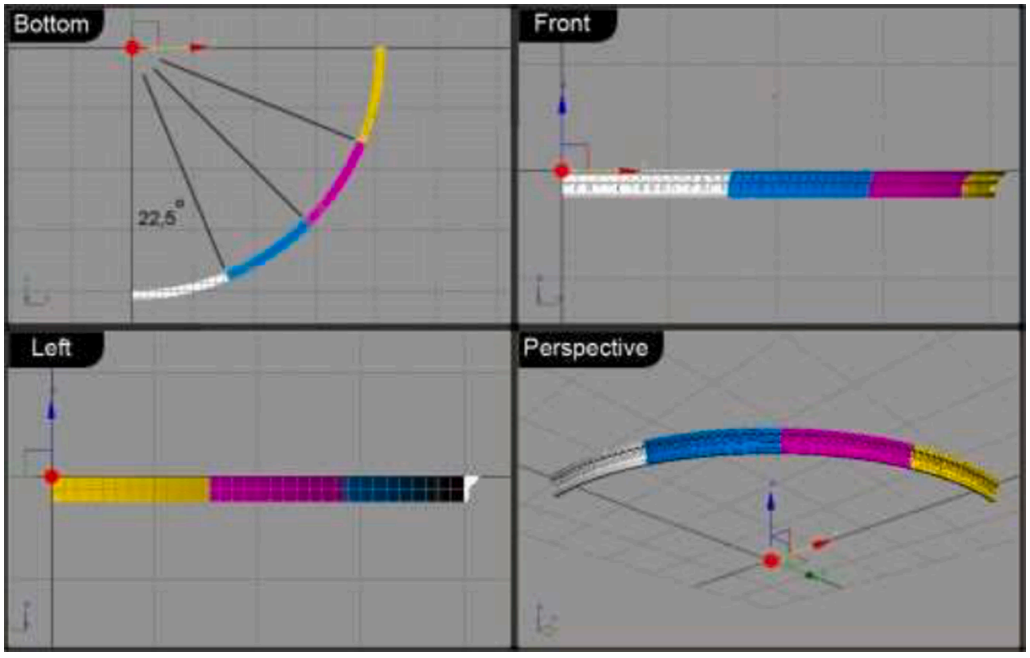


Fig. 3. Radial modular meshes benefit from placing the pivot point at the centre of rotation of the mesh, as demonstrated in this example (Ibid.).

City building games also benefit when limited to multiples of the grid size. Building sizes of 2x2, 2x4, 4x4, and 8x8 for example are easier to pack within a city block and can maximize the built space. An irregular building of 1x3 would be harder to pack within the same city block and tend to leave empty spaces.

4.4. Grid space

When designing modular assets, it is important to stay completely within the asset's assigned grid space, as any elements that exceed the grid space may overlap other meshes during level design. Thus, the assigned size of an asset is always the *total* grid space, and not necessarily the usable floorplan. For example, a corridor can be designed with plain flat walls, maximizing the usable floorplan. But if a corridor is designed with elaborate concave walls and decorative frames, the usable floorplan will be considerably smaller [2].

4.5. Pivot point

All 3D meshes have a pivot point from which they can be moved, rotated or scaled. It is important that all pivot points follow a common logic, placing the pivot point at a consistent location that facilitates the

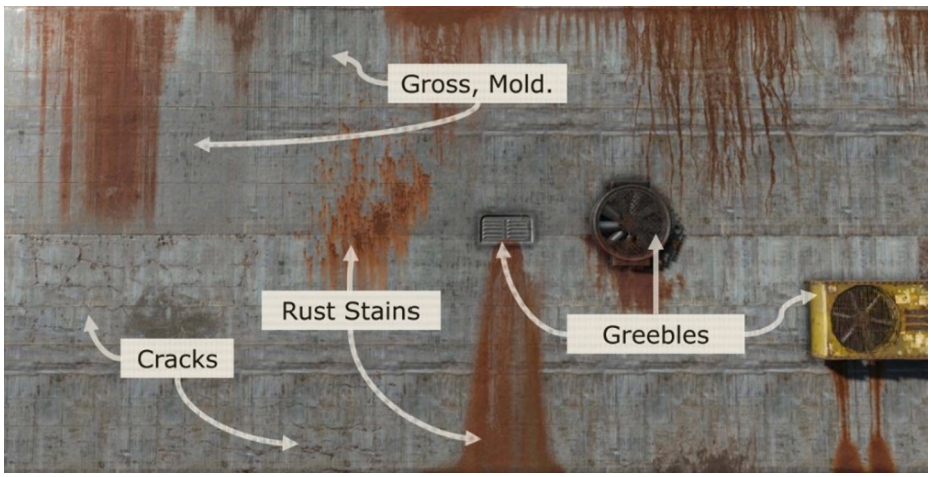


Fig. 4. Examples of decals and greebles [2].

alignment or tiling of multiple meshes within the engine [19]. For modular meshes and props with horizontal tiling, it is common for the pivot point to be at the bottom corner or top corner (always the same right or left corner) as in Fig. 1.

For corner modular meshes, modular meshes with a "round" profile (such as a pillar), and non-tileable props, it is common for the pivot point to be at the bottom centre (Fig. 2).

For radial modular assets, such as segments of a donut or round platform, placing the pivot point at the centre of rotation can be the most effective solution, even if the pivot point is technically outside the mesh space, as in Fig. 3.

4.6. Naming convention

A clear and well-documented naming convention is fundamental for game assets. As names are likely to get long, using abbreviations or omitting vowels are standard. The authors suggest the following naming convention for modular environment assets, which can be adapted as needed: *StaticMesh*[2]*_NameOfKit_TypeOfAsset_Dimension_Characteristic_Variation*. For example: *SM_MdInt_Wll_S_Ofc_DmgTL* (StaticMesh_ModernInterior_Wall_Small_ Office_DamageTopLeft).

4.7. Modular kit

At the heart of modular architecture for games is the modular kit. While not all modular assets are designed as kits, the time invested in planning a modular kit pays off in a well-organized production with a lower need for rework or unforeseen new assets [2]. On the other hand, when a game development approaches modular architecture without extensive planning and organization of assets into modular kits before production starts, the final collection of assets often includes duplicates, lacks standardization, and requires a large number of trims and make up assets. While this might be acceptable on a small game, on large and open world games such approach is rarely cost-effective [17].

Typical elements of a modular architectural kit include:

Wall panels: Interior and exterior wall panels are usually independent and tend to be one-sided, except for transition points [19].
Transition points: Include doorways, windows, archways, and other points that connect different locations and/or kits. It can be advantageous to cap transition points with a plug and socket system using independent meshes, similar to door frames in traditional architecture [2,13,17].
Floors and roofs: Similar to wall panels, interior, and exterior floors and roofs are usually independent and tend to be one-sided. In an interior environment, floors and roofs can be combined for multi-level structures.
Corridor modules: Corridors are usually built as volumetric modules, either directly exported as such from the 3D modelling software or pre-assembled via prefab in the game engine. The volumetric corridor slices (including floor, walls, and ceiling as a single module) must account for the layouts needed for the game and include the necessary T or square junctions.
Trims: Independent meshes that can hide gaps and overlaps between assets or cap one-sided meshes such as the sidings of a raised foundation or the corners of exterior walls. When well designed, trims can enhance the architectural style by adding pillars, columns, quoins, wood, or metal frames [2,13].
Vertical transitions: To move vertically within levels, games rely on similar transitions as traditional architecture: stairs, ladders, ramps, and lifts. Ramps can be part of the architectural design or sunken floors and debris in the case of ruins [2].

The following elements are also part of the kit, and can often be shared between multiple kits:

Shared components: Independent elements that fit into standard fittings in multiple modular kits, for example, doors, windows, porches, chimneys, and visible light fixtures [2,17].
Movable elements: Elements that move either via driven animation or physics need to be independent meshes. For example doors that open, chandeliers, levers, buttons, gears, or parts of puzzles [13,19].
Set dressing assets: These can be specific to kits or general props. Set dressing designed for modular kits include elements that indicate functionality – even if they are not actually interactive – such as control panels, modular pipe, and wiring. They can also include furniture designed for a specific environment, as the interior of a cafeteria or a library [2,17].
Decals and greebles: Overlapping meshes that add extra detail and variations to a surface (Fig. 4). Decals are non-volumetric meshes that add detail through texture variation (usually with transparency mask), commonly used to add localized rust, mould, moss, graffiti, logos, posters, or bullet marks. Greebles (also known as nurnies) are meshes with low-volume that add localized 3D components to surfaces, such as manholes, plug sockets, ventilation grids, intercoms, etc. [2,21]. Decals and greebles are usually shared throughout the

[2] In game development, static meshes describes game assets that are not intended to be deformable or interactive, and are faster and cheaper to render in real-time. Environment meshes tend to be static meshes by default.

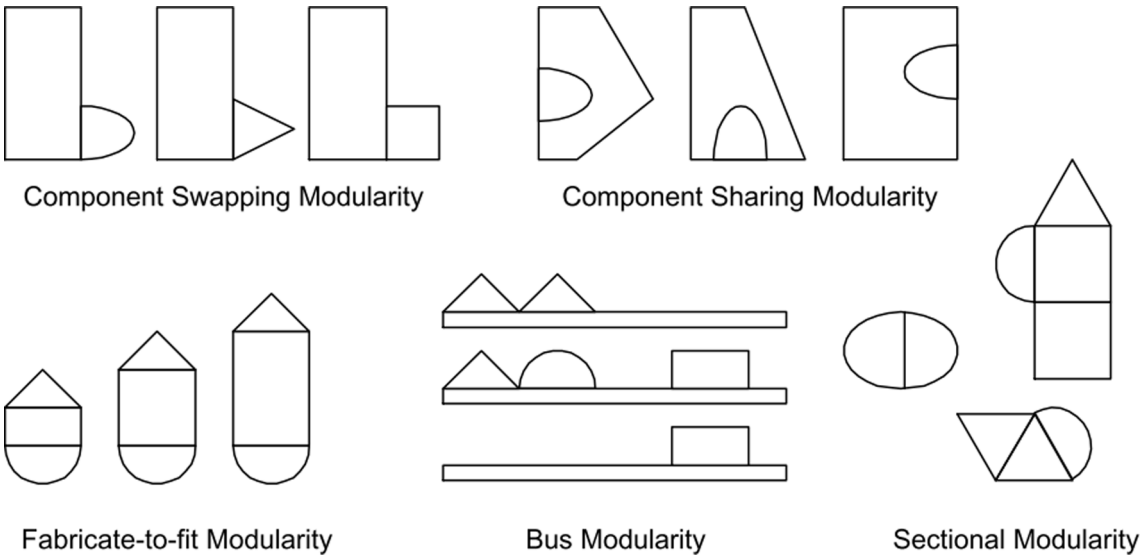


Fig. 5. Types of modularity, derived from mass customization [24].

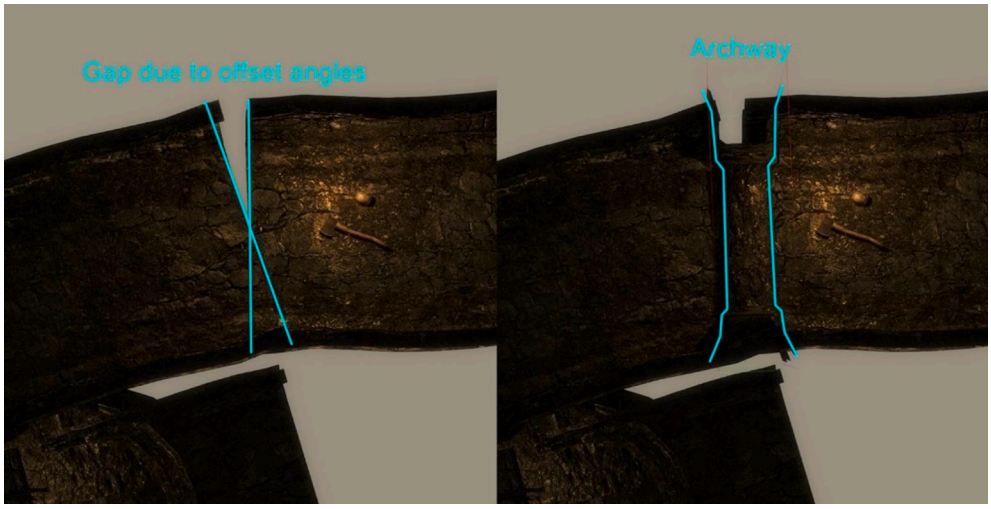


Fig. 6. Filler meshes are needed to create irregular pathways that break the grid, and may be designed as fabricate-to-fit modularity [20].

game across multiple kits, and are added by level designers during the visual polish phase.

4.8. Types of modular kits

Large scale games may require several kits to organize the variety of architectural styles, construction techniques, and factions. During pre-production, planning should consider the needs of the game: what kind of construction techniques, types of spaces, factions, and time periods are represented? How do those interact together? What are unique builds, and what are only variations?

As a rule of thumb, fewer kits with more components tend to be more cost-effective than more kits with fewer components. Each kit, however small, requires a unique logic of how the assets are assembled together which demands careful planning, design, and functionality test, a process that can be very time consuming. Therefore once a working kit logic is in place, it is generally easier and faster to expand that kit than to create a new kit with a new unique logic instead. That is not to say that kits should be forced together: there is little benefit for example of forcing a kit for a large cathedral into that of a residential unit. However, it is possible that modern offices and retail can benefit from sharing a common kit. Determining the types of modular kits a game needs is an ongoing process that is likely to be consolidated or splintered as development progress [2,17].

4.9. Types of modularity

Common types of modularity featured in architecture include sectional, component sharing, component swapping, fabricate-to-fit, mix and bus modularity [13], exemplified in Fig. 5. What type of modularity to use will influence how the assets are designed; most games rely on sectional, component swapping, and component sharing modularity [22,23]. Regardless of the types of modularity adopted, standardization is key to maximize asset combination and consequently visual variation.

Sectional modularity: This is the most common type of modularity in games, relying on assets being tiled horizontally, vertically, or both. It relies on assets sharing the same dimensions and also being visually continuous. Sectional modularity is used to tile walls, floors, roofs, facades, and corridors [19].

Component sharing: This type of modularity relies on the same component being used by multiple elements. Common examples are

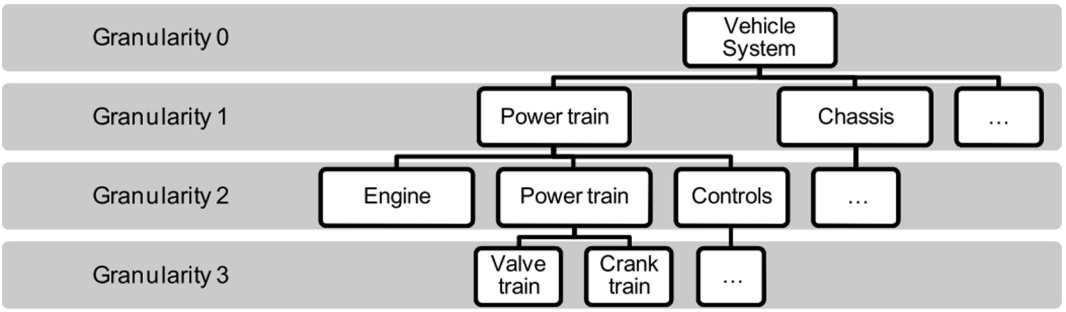


Fig. 7. Example of partial levels of granularity of a vehicle system, according to system development [25].

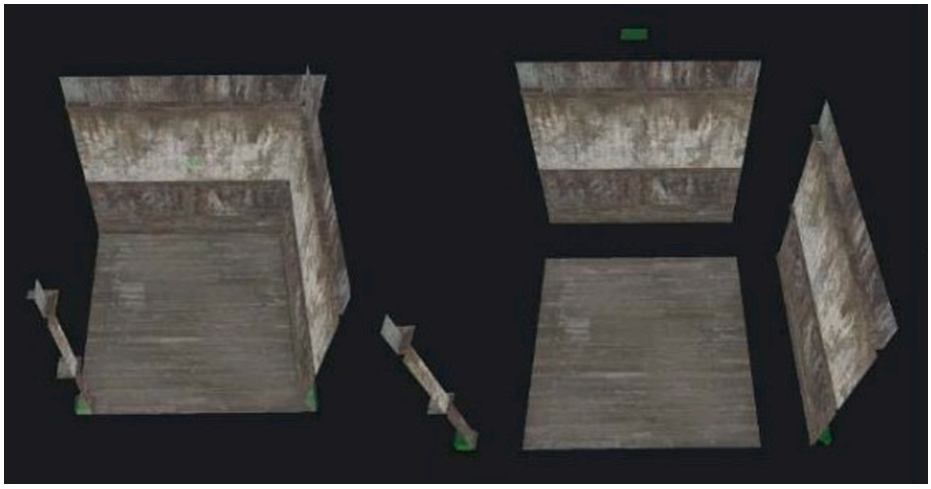


Fig. 8. Left: pre-assembled module in engine as a volumetric module via prefab. Right: all the meshes are exported as individual components [2].

windows and doors, where a limited range of both are shared by multiple types of walls and facades. Sometimes these components are shared by multiple kits [17].

Component swapping: When different components can fit into the same slot. This can be useful to create variations, for example different caps for the same column base can quickly add variation while minimizing the total number of assets [19].

Fabricate-to-fit: Meshes that can be scaled to bridge gaps between two other meshes. Filler meshes such as Fig. 6 may be designed following fabricate-to-fit modularity.

Mix modularity: This is a technique that can add more variations with fewer assets, but tends to be underused in games. For example, a full-sized wall can be split between a bottom and a top section, where the bottom can be wood panels, stone or different colours, while the top can be wallpaper, textured plaster or different colours. A decorative statue can be made up of a bottom support that is separate from the actual statue, so those can be randomly combined increasing the number of possible variations [19].

Bus modularity: This type of modularity relies on a supporting frame to which multiple modules can be attached. This is typically used to create large modular structures such as spaceships, space stations [8], and also in city building games where special modules or expansions can be added to a base structure.

4.10. Tiling direction

Sectional modularity, where assets are tiled side by side, is the most common type of modularity in games. Here it is important to determine the tiling direction of the meshes: will they support horizontal modularity, vertical modularity, or both. It is important to consider how these elements will be visible to the player, and if trim meshes are needed to cap visible gaps at the corners for example [20].

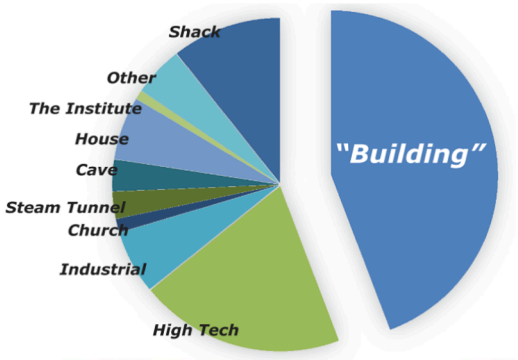
4.11. Level of granularity and prefab

Broadly speaking, both traditional architecture and Bethesda's guidelines approach volumetric architectural meshes as low granularity and non-volumetric meshes as high granularity. Thus a room or pre-assembled section of a room are considered low-granularity modular meshes, whereas individual wall, floor and ceiling segments are considered high-granularity [2,10].

System development, where level of granularity is also known as architectural decomposition, encourages a finer distinction between levels of granularity as exemplified in Fig. 7. Here the levels of modularity are used to reduce the complexity of a system into manageable parts [25]. This is a good analogy for modular assets for game environments, where the level of granularity varies depending on the type of game, as will be discussed next. Also similar to system development, high granularity modular environment assets are nested into increasingly larger and more complex groups which eventually form the full architectural structure at the lowest level of granularity, or level 0 as described by system development. This process of grouping high modularity assets into lower granularity can be referred to in game development as prefab and is an intermediate step that groups assets in engine prior to level design to facilitate level assembly [2,26].

Modular kits that rely on smaller granularity (smaller individual meshes) tend to be more flexible and offer greater visual variation. Changes to assets are generally easier to implement, requiring a single re-export of the individual mesh. On the other hand, when assembling the levels within the engine, too many small assets can be fiddly and generally require a longer assembly time per level. Each of the assets must be perfectly aligned to avoid visible gaps and overlaps, and the

Table 4
Principles of modular architecture for games according to a typical game development cycle.

Step	Description	Example/ Further 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 [19,20].	A single game may be developed by studios in different countries where different measuring units are the norm. A unified measuring convention must be adopted for the whole game.
Determine the grid size	Game engines rely on Cartesian grids, and modular architecture is built upon multiples of a minimum 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 [2,19].	The environment art grid size can impact several areas of the gameplay, including character navigation, movement range, camera position and FOV, and affects the level design and game mechanics. In Fig. 9 the minimum grid size is 3 m or 1.5x the character height. In Fig. 10 the minimum grid size is 4 m or 2x the character height[1].
	 Fig. 9: grid size 3 m.	 Fig. 10: grid size 4 m.
Establishing 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.	A good naming convention should be descriptive and standardized. For example: StaticMesh_ NameOfKit_TypeOfAsset _Dimension_Characteristic_Variation.
Define the types of modular kits	Large-scale games may require several kits to organize the 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].	For example, in the game Fallout 4 the types of modular kits were defined by architectural features and included a main building kit and kits for other types of environments such as church, high-tech, industrial, steam tunnel, cave, house, shack, the institute, and other. Fig. 11 displays a breakdown of time allocation per type of modular kit for the game.
	 Fig. 11: Modular kits in Fallout 4 organized by development time [2].	
Define the types of modularity	What type of modularity to use will influence how the assets are designed; most games rely on sectional, component swapping, and component sharing modularity [19].	Most games rely heavily on sectional modularity, with both horizontal and vertical modularity to create large grids of repetitive assets. Other types of modularity such as component swapping and component sharing for example can help to add visual variation.
Pre-production		
Whiteboxing	As with non-modular level design, pre-production starts with whiteboxing [18], 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 [2,19].	At this stage level designers and environment artists should collaborate to define types of level needed for the game, and the types of modular environment assets needed to build each type of level. A desirable outcome are playable whitebox levels and preliminary lists of modular assets and modular kits [17].
Refine the 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].	As fewer kits with more assets are generally less time consuming to produce than many kits with few assets each, it may be advisable to consolidate modular kits where possible to optimize production. This is an organizational step that may involve quick prototype tests and refinement of the list of modular assets [2,17].
Plan the modular kits	This involves several sub-steps described from (a) to (h).	At this stage the list of assets should be finalized as much as possible prior to start of production, facilitating time planning and allocation of resources. Technical conventions such as the position of pivot points and transitions should be detailed with the aid of clear documentation.
	(a) assets per modular kits	Create a detailed list of assets per modular kit. While variations in style and decoration may be added later, all the types of basic assets should be listed [17].
	(b) tiling direction	Sectional modularity, where assets are tiled side by side, is the most common type of modularity in games [19]. 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].

(*continued on next page*)

Table 4 (*continued*)

Step	Description	Example/ Further Description
	(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 [19].
	(f) standard size of transitions	Different kits can be connected seamless by adopting standard sizes for transitions such as doorframes and stairs [2,13].
	(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 [2,13].
	(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 [2,17].
Grey box	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 ingame testing and iteration [17].	The greybox meshes should follow all the conventions previously established, including correct naming, size and use of pivot point. At the end of this process, levels should be playable and form continuous visual environments without gaps or misaligned modular assets [17]. Greyboxes may highlight the need for further assets or for corrective modular assets such as caps for transitions.
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].	This step should result in a functional modelled and textured version of each kit, although polishes may still be lacking. The vanilla variation should enable the various level design configuration needed in the game, although with little visual variation [17].
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].	Examples of functional polishes include correction of lack of modular continuity between assets, gaps between modular assets, incorrect size transitions or lack of assets. Any visual errors or problems with the modular kits and modular assets should be addressed before continuing production.
Production		
Create and implement variations	This stage focuses on adding variations based on the basic/vanilla version, usually combining mesh and texture variations [2].	Examples of variation include different surface finishes, decoration and damages. These may affect both the mesh and texture or only the texture. Fig. 12 shows an example of variations for Fallout 4's Building modular kit, with highlighted brick variation.

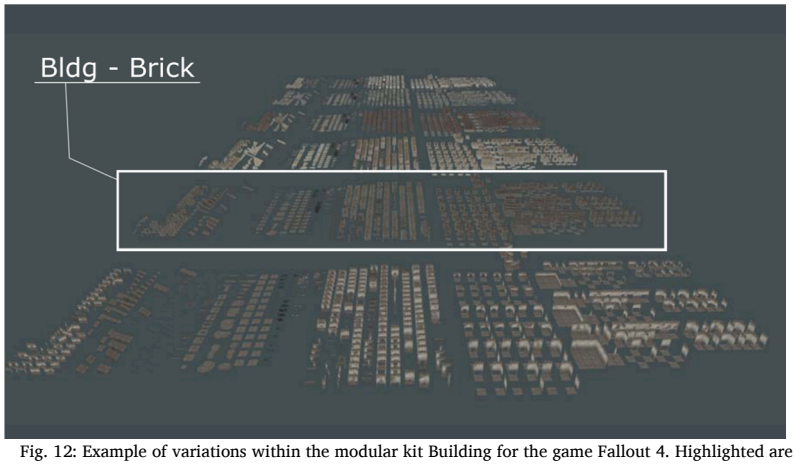


Fig. 12: Example of variations within the modular kit Building for the game Fallout 4. Highlighted are the assets for the variation brick [2].

Create and implement set dressing and props	Set dressing components and props are also added. These might be modular or stand-alone assets [2].	Examples of set dressing are small decorations, plants, rugs, books, small electronics and appliances, rugs, curtains, etc. Good set dressing can help to minimize visual repetition within modular levels.
Polish		
Visual polish	Previously identified visual weaknesses should be addressed. New meshes and textures should generally not be added at this stage [2].	At this stage production of new assets should generally be concluded. It is recommended that this step be reserved for final touch ups and correction of visual bugs, with a focus on environment art meshes and textures. These items may have been noticed during quality assurance or by lead artists.
Optimization	Standard game graphic optimizations such as packing texture maps, adjusting LODs, and graphic quality balancing [2].	Optimization may include collaboration between environment artists and technical artists and/or programmers. The goal is to ensure a smooth game playback by reducing the texture size, adjusting the mesh resolution based on camera distance, baking lightmaps, etc.
Correcting bugs	Previously identified bugs should be corrected.	Any other bugs identified during quality assurance or revisions should be addressed at this point. These may include visual or technical issues and may require collaboration between environment artists, level designers and technical artists to identify the cause of the bug.

[1] A. Galuzin, UE4: Guide to Player Scale and World Architecture Dimensions, World of Level Design. (2015). https://www.worldofleveldesign.com/categories/ue4/ue4-guide-to-scale-dimensions.php (accessed September 12, 2021).

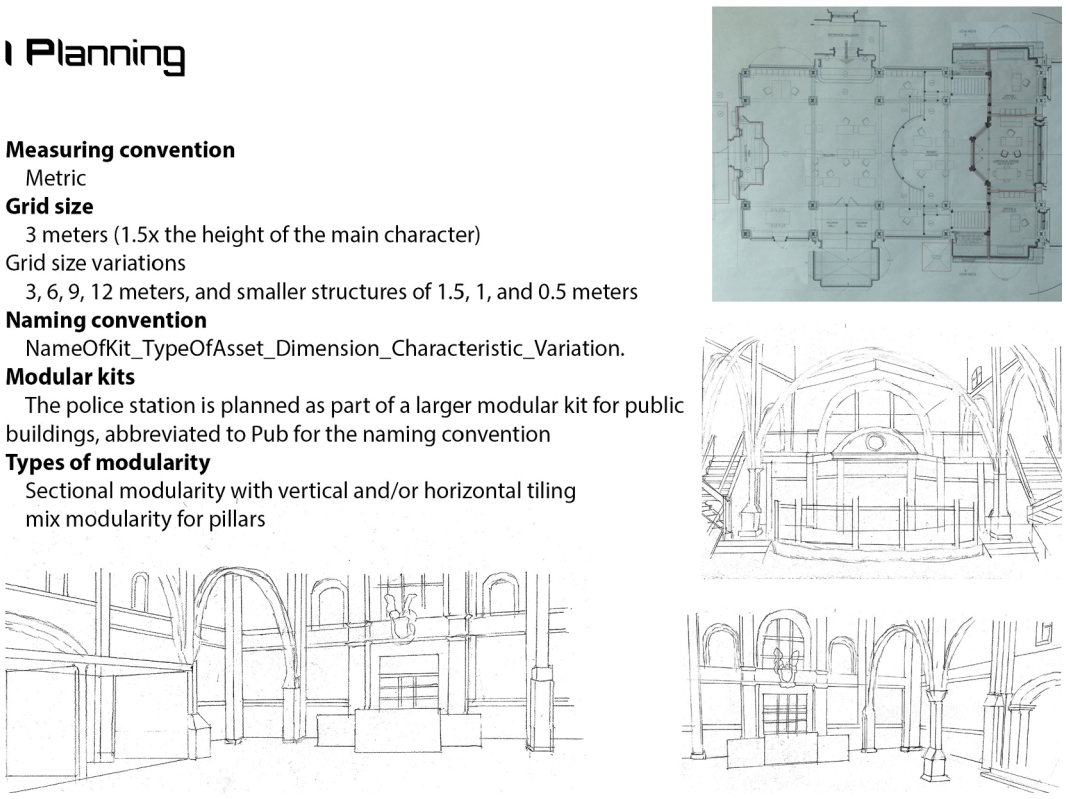


Fig. 13. Overview of proof-of-concept prototype with focus on the Planning step.

more granular they are, the more likely mistakes are [2].

Larger, more pre-assembled assets can speed up the level assembly, but they limit the flexibility of the kit. Pre-assembled assets are exported from the 3D modelling software as single meshes, often combining smaller meshes within the 3D software. While this process facilitates level assembly by keeping the number of components within the engine manageable, it hinders changes in the meshes. If a late change requires a mesh detail such as the height of a window needs to be raised, all the assets that feature this window must be individually adjusted and re-exported from the 3D software. Using smaller granularity, only a single mesh with the wall and window segment would need to be adjusted and re-exported [2].

A solution to balance the flexibility of smaller granularity with the ease of assembly of larger assets is to use the prefab or pre-assembly feature offered by some game engines. This allows meshes to be exported from the 3D modeling software using small granularity, and to be grouped through an intermediate step within the game engine into larger prefabs before being added to the game level, as exemplified in Fig. 8 [2]. This step also allows lighting and functional elements such as collision boxes to be added to the prefab, further optimizing the level assembly.

5. Designing and implementing modular architecture for games

There are several steps required to design and implement modular architecture for a game. Table 4 describes the principles proposed in this paper by development phase, following a typical game development cycle with a focus on environment art and level design.

5.1. Planning

Before production starts, the team needs to decide the fundamental characteristics of the modular environment [18]. Generally this process involves the leads and management: lead environment artist, lead level designer, producer, etc. Consultation with an architect or industrial designer can be beneficial to clarify the architectural style(s) and visual characteristics of the modular kit. During planning, the following should be decided and documented: the measuring convention, the grid size and allowed variations, the naming convention, the types of kits, and the types of modularity [19]. While the measuring system and grid size should ideally not change, the types of kits and types of modularity are likely to evolve and be reiterated during pre-production.

5.2. Pre-production

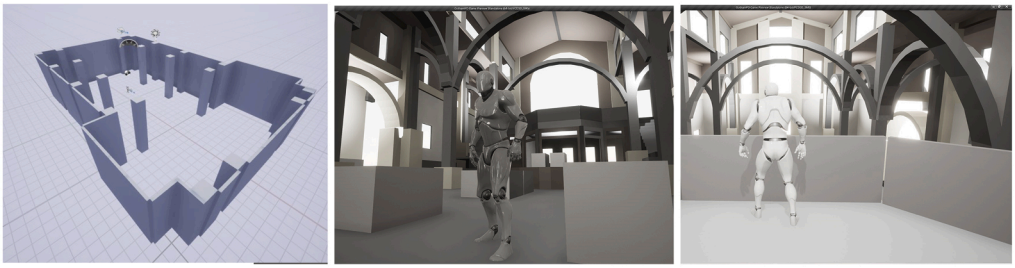
This is the research and development phase, where the focus is to establish how the modular kits will function before there is a high artistic investment. This process is iterative and involves close coordination between level designers and environment artists. Level design starts with whiteboxing [17], 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.

Once the modular kits have been planned, environment artists create rough placeholder grey meshes (untextured) with the correct dimensions and layout to test the functionality of the kits [2]. These placeholder meshes replace the whiteboxing meshes, allowing both artists and level designers to test them in-game to verify their functionality and quickly iterate as needed [17]. During this process it should be established:

- what basic assets are needed, their dimensions and characteristics,
- the type of modularity and tiling direction of each asset,
- the level of granularity of the kit,
- whether to use prefabs or not,
- where to position pivot points,
- the standard size of transitions,
- whether transitions are plug and socket or special meshes,

2 Pre-production

2.1 Whitebox



2.2 List of Assets

Name	Size	Variations	Tiling
Walls			
Pub_Wll_DD_Basic	6x6	no windows	horizontal + vertical
Pub_Wll_DD_Window	6x6	single window	horizontal + vertical
Pub_Wll_DS_Basic	6x3	no windows	horizontal + vertical
Pub_Wll_DS_2Windows	6x3	double windows	horizontal + vertical
Pub_Wll_DS_Window	6x3	large window	horizontal + vertical
Pub_Wll_DS_3Windows	6x3	3 office windows	horizontal + vertical
Pub_Wll_SS_Basic	3x3	no window	horizontal + vertical
Pub_Wll_SS_GlassDoor	3x3	office door	horizontal + vertical
Pub_Wll_SD_Basic	3x6	no window	horizontal + vertical
Pub_Wll_SD_Window		single window	horizontal + vertical
Pub_Wll_HS_Basic	3x1.5	no window	horizontal + vertical
Pub_Wll_HS_GlassDoor		small glass door	horizontal + vertical
Ceilings and Floors			
Pub_Clg_DD_Basic	6x6	plain ceiling	horizontal + vertical
Pub_Clg_SD_Basic	6x3	plain ceiling	horizontal + vertical
Pub_ClgFlr_DD_Basic	6x6	ceiling + floor	horizontal + vertical
Pub_ClgFlr_SD_Basic	6x3	ceiling + floor	horizontal + vertical
Pub_Flr_DD_Basic	6x6	floor	horizontal + vertical
Pub_Flr_SD_Basic	6x3	floor	horizontal + vertical
Trims			
Pub_Trim_S_Corner	3x.25x.25	plain	horizontal
Pub_Trim_S_Wall	6x.25x.5	flat, against wall	horizontal
Pub_Railing_S	1x1x.1	railing	horizontal

2.3 Greybox

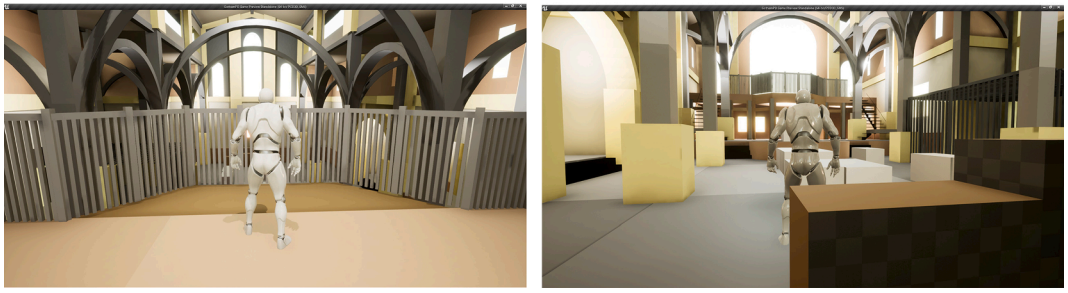


Fig. 14. Overview of proof-of-concept prototype with focus on the Pre-Production step.

- what assets can be shared between kits,
- the standard sizes of stairs, ladders, seats, covers, and other interactive assets.

5.3. Early production

If the previous steps have been well-established, early production focuses on the development of the basic components of the kit from placeholder geometry to final meshes. As long as the name and the pivot point position are unchanged, replacing placeholder meshes with final ones is a simple process of re-exporting the meshes to engine.

Usually production follows a priority list, where the basic components of each kit tend to have higher priority and are usually developed first, so there will be at least one example of each type of component in the game. At the same time, level designers can start building the game's levels using the placeholder geometry. At this stage, it is likely that the

Fig. 15. Comparison greybox and TV show reference: chancel seen from the top mezzanine.

team will identify assets that are missing or that are redundant within the kits. The consolidation of modular kits or adding new types of kits is also common in early production [2].

The focus of early production is to confirm the design of the kits through a first and basic 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. Small changes, such as adding an extra type of component, can be fast and easy to implement at this stage. Overarching changes, such as changing the grid size, will require changing most of the individual assets and can be time consuming. However, it is best to identify the need for such changes at this early stage with only the basic version than later in production, when such changes become prohibitively expensive and likely to result in production delays and over-budgeting.

Early production should finish with a functional polish of the kits, where the basic variations are in engine replacing the placeholder versions in the levels already developed. While visual polishes may still be needed during the Polish stage, it is important to ascertain that the kits and the basic assets of each kit are functionally sound before investing time and resources into creating variations [2].

5.4. Production

This stage focuses on adding variations and creating the set dressing and prop assets. This is when the vast majority of environment assets will be created, based upon the basic variation tested in early production. Variations are usually developed following a priority list, and as more variations of each kit become available, levels can be updated to incorporate them randomly, manually, or by a combination of both. As AI and machine learning are further integrated into games, it is likely that this process will become further automated, with only a final manual pass required [27].

During production, set dressing and props are also produced and incorporated into the levels. Any changes to the structure of the kits during production becomes expensive, and changes to the grid size basic dimensions is generally prohibitively expensive [2]. With good planning and testing during the earlier phases, at this point kits should generally only require adding a handful of extra types of assets or variations to accommodate unforeseen needs in levels as the game is further developed.

5.5. Polish

This stage includes visual polish, optimizations, and correction of bugs. It can vary depending on the studio and the amount of production time reserved for polish. This can involve correcting previously identified bugs or visual weaknesses, packing texture maps together, adjusting LODs, and other common optimization procedures. New meshes and textures should generally not be added at this stage.

5.6. Proof of concept prototype

The proposed principles were exemplified with a proof of concept prototype based on a transmedia location featuring the interior of the

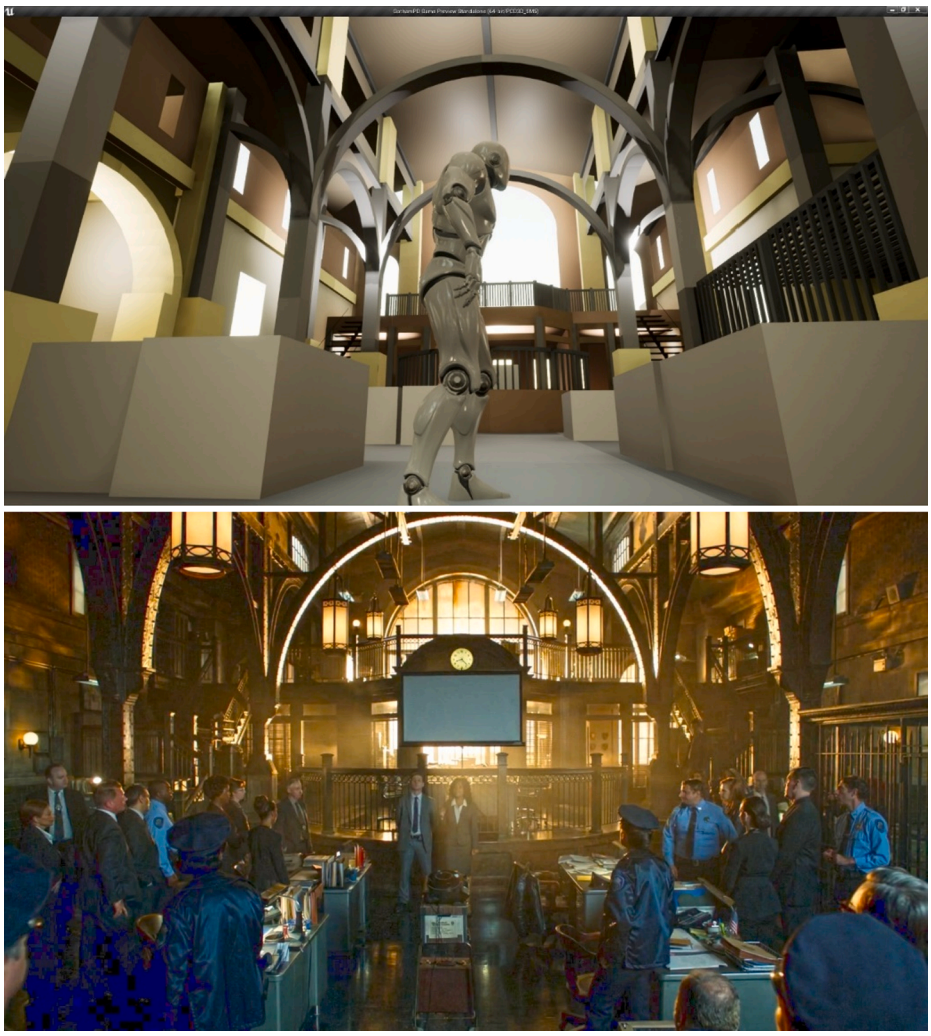


Fig. 16. Comparison greybox and TV show reference: nave and structural arcs.

Gotham City Police Station as depicted in the TV show Gotham aired between 2014 and 2019. It was developed using Unreal Engine 4 release 4.24 and Autodesk 3ds Max 2020.

The prototype focused on the steps of planning and pre-production to demonstrate the potential of the proposed principles in aiding the design of game environment art with modular architecture. Figs. 13 and 14 summarize the principles described in section 4 applied to the prototype, with a focus on the steps of planning (Fig. 13) and pre-production (Fig. 14). The table within Fig. 14 describes a partial list of assets for the prototype, which totalled 42 modular architectural meshes, not including props and set dressing.

By following the principles and production steps it was possible to translate a complex pre-existing location into a game environment based upon modular architectural meshes. These meshes are individually simpler than recreating the location as a single hero mesh, and most importantly, they allow a great degree of flexibility of layout and reusability of assets. Figs. 15 and 16 compare the greybox version with the reference from the TV show, demonstrating the potential that modular assets have of recreating complex architectural locations when basic principles are observed.

6. Conclusion

The principles proposed here aim to fill an important knowledge gap in describing and documenting modular architecture for 3D game environment art. The authors hope that these unified principles combined with suggestions for how to design and implement modular kits within a game development cycle may guide game developers to incorporate modularity effectively while minimizing implementation problems. It is important that modular architecture is adopted as a formative part of the game's environment design and visual aesthetic, and not as an afterthought. Close communication between level designers and environment artists is key to successful modular architecture, and both teams must be closely involved with the planning and design of the modular kits for the game [2,17].

Each game is unique and the specific needs of the game will drive decisions. For example, the series *Assassin's Creed* was unique among the studied games in requiring all exterior architecture to be explorable, placing a higher level of importance on the facades and rooflines of buildings. It is important that the design of the modular architecture takes into consideration and facilitate the desired gameplay. Similar types of games share similar modular characteristics, as reviewed in section 2, and key characteristics such as regular grids, a defined grid size, and careful placement of the pivot point are essential to enable modularity [2,19]. On the other hand, the types of modularity, level of granularity, and how to organize the modular kits can vary considerably from game to game.

A fundamental difference in relying on modular architecture versus unique hero assets is that modularity requires extensive and early planning. With hero assets, new locations can be added as needed, but with modular assets decisions regarding the types of locations must be made early, before actual game production starts [23]. Adding new

types of locations or layouts can be hard and expensive, usually resulting in production delays. On the other hand, well-designed modular kits can enable smaller teams to populate large game maps in a cost-effective manner, optimizing both human and engine resources [2]. The inherent repetition of modular architecture can be used by skilled level designers as a positive design feature, accentuating the architectural style. It can also facilitate the integration of user generated content [17].

Set dressing and a large enough number of variations are key to minimize repetition and art fatigue, and here developers can benefit from automation. Upcoming tools such as Promethean AI use artificial intelligence to customize sections of a game level based on a written description such as a "teenage boy room" which automatically fills the space with the adequate props and set dressing. Designers can further tweak it by adjusting the level of tidiness of the room[3]. Modelling and texturing can also be automated or aided via procedural generation and photogrammetry, where further development of tools and techniques is still in development. Investment in automation tools and processes combined with effective implementation of modular architecture for games can be the key to handling the trend of ever larger and more complex game maps while keeping art production manageable.

The authors recommend that the principles proposed here be used as a flexible guideline by developers, understanding that different projects will have different needs and requirements. The constant improvement in the tools and technologies involved in game development also requires that these principles be periodically revised and tested by developers and researchers.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Greuter, A. Nash, Game asset repetition, in: ACM Int. Conf. Proceeding Ser., Association for Computing Machinery, New York, New York, USA, 2014, pp. 1–5. https://doi.org/10.1145/2677758.2677782.

[2] J. Burgess, N. Purkeypile, "Fallout 4's" Modular Level Design, in: Game Dev. Conf. GDC 2016, InformaTech, San Francisco, CA, US, 2016.

[3] E. Adams, Fundamentals of Game Design, third ed., New Riders, Berkeley, CA, 2013.

[4] H. Jenkins, Game design as narrative architecture, in: N. Wardrip-Fruin, P. Harrigan (Eds.), First Pers. New Media as Story, Performance, Game, MIT Presss, Cambridge, MA, USA, 2004, pp. 118–130.

[5] T. Heussner, The Advanced Game Narrative Toolbox, CRC Press Taylor & Francis Group, Boca Raton, FL, USA, 2019.

[6] F. Rotzetter, Nonverbal guidance systems seamless player-leading in open-world games, in: B. Suter, M. Kocher, R. Rauer (Eds.), Games Rules: Game Mechanics for the "Magic Circle", ZHdk Zurich University of the Arts, 2018, pp. 169–190.

[7] M. Wellenreiter, Screenwriting and authorial control in narrative video games, J. Screenwriting 6 (3) (2015) 343–361.

[8] J. Esper, Modular, adaptive, reconfigurable systems: technology for sustainable, reliable, effective, and affordable space exploration, in: AIP Conf. Proc. 746, Albuquerque, New Mexico (USA), 2005, pp. 1033–1043. https://doi.org/10.1063/1.1867227.

[9] B. Bereitschaft, Gods of the city? Reflecting on city building games as an early introduction to urban systems, J. Geog. 115 (2) (2016) 51–60, https://doi.org/10.1080/00221341.2015.1070366.

[10] AIA, NIBS, Design for Modular Construction: An Introduction for Architects, New York, NY, USA, 2019.

[11] T. Linner, T. Bock, Evolution of large-scale industrialisation and service innovation in Japanese prefabrication industry, Constr. Innov. 12 (2) (2012) 156–178, https://doi.org/10.1108/14714171211215921.

[12] I. Johansson, Defining what is visually dynamic for modular assets in level design, Technical University of Lulea (2017).

[13] R.E. Smith, Prefab Architecture: A Guide to Modular Design and Construction, John Wiley & Sons, Hoboken, New Jersey, 2010.

[14] T. Bock, S. Langenberg, Changing building sites: industrialisation and automation of the building process, Archit. Des. 84 (2014) 88–99, https://doi.org/10.1002/ad.1762.

[15] M. Luther, L. Moreschini, H. Pallot, Revisiting prefabricated building systems for the future, in: ANZASCA 2007 Proc. 41st Annu. Conf. Towar. Solut. a Liveable Futur. Progress, Pract. Performance, People, Deakin University - School of Architecture & Building, Geelong, Australia, 2007, pp. 157–164.

[16] M. Aitchison, A house is not a car (yet), J. Archit. Educ. 71 (1) (2017) 10–21, https://doi.org/10.1080/10464883.2017.1260915.

[17] M. Salmond, Video Game Level Design: How to Create Video Games with Emotion, Interaction, Bloomsbury Academic, London, UK, 2021.

[18] W. Suanto, Y.S. Martyastiadi, Modular technique of 3D modeling and procedural texturing for 3D game environment design of "Jurnal Pahlawan," in: Int. Conf. Intermedia Arts Creat. Technol., Yogyakarta, Indonesia, 2020, pp. 5–12. https://doi.org/10.5220/0008525200050012.

[19] L. Perry, Modular level and component design, Game Dev. (2002) 30–35.

[20] J. Burgess, N. Purkeypile, Skyrim's modular level design, in: GDC 2013 Game Dev. Conf., InformaTech, San Francisco, CA, US, 2013.

[21] K. Kinnear, C.S. Kaplan, Procedural generation of surface detail for science fiction spaceships, in: Proceedings of the Sixth international conference on Computational Aesthetics in Graphics, Visualization and Imaging, in: Comput. Aesthetics'10 Proc. Sixth Int. Conf. Comput. Aesthet. Graph. Vis. Imaging, 2010, pp. 83–90. https://dl.acm.org/doi/10.5555/2381312.2381327.

[22] S. Jones, Investigation into Modular Design within Computer Games, Staffordshire University, 2011.

[23] N. Statham, J. Jacob, M. Fridenfalk, R. Rodrigues, Recreating a TransMedia architectural location in-game via modular environment assets, in: J. Baalsrud Hauge et al. (Ed.), IFIP-ICEC 2021 20th IFIP Int. Conf. Entertain. Comput., Springer Nature Switzerland AG 2021, Coimbra, Portugal, 2021, pp. 1–9. https://doi.org/10.1007/978-3-030-89394-1_29.

[24] T.D. Miller, P. Elgård, Defining modules, modularity and modularization: evolution of the concept in a historical perspective, in: Des. Integr. Manuf., Fuglsoe, Denmark, 1998, pp. 1–19.

[25] N. Chiriac, K. Hölttä-Otto, D. Lysy, E.S. Suh, Level of modularity at different levels of system granularity, Proc. ASME Des. Eng. Tech. Conf. 9 (2011) 329–339, https://doi.org/10.1115/DETC2011-47889.

[26] J.S. Ortiz, B.S. Guevara, E.G. Espinosa, J. Santana, L.R. Tamayo, V.H. Andaluz, 3D virtual content for education applications, in: J.S. Ortiz, B.S. Guevara, E. G. Espinosa, J. Santana, L.R. Tamayo, V.H. Andaluz (Eds.), Iber. Conf. Inf. Syst. Technol. Cist., IEEE Computer Society, Seville, Spain, 2020, pp. 1–5, https://doi.org/10.23919/CISTI49556.2020.9140822.

[27] M. Guzdial, N. Liao, M. Riedl, Co-creative level design via machine learning, CEUR Workshop Proc. 2282 (2018). https://arxiv.org/abs/1809.09420v1.

[3] A. Maximov, Promethean AI: The Tricks of Learning Game Art, 80 Lev. (2018). https://80.lv/articles/promethean-ai-the-tricks-of-learning-game-art/ (accessed March 24, 2020).