# Libre Gaming Manifest

Present and Future

pyramid contributors

release date: 2020-05-11

## Contents

```
#
# @file   : libre-gaming-manifest.md
# @version: 2020-05-11
# @created: 2019-02-01
#
```

# 1 Introduction

**Libre Gaming Manifest** (LGM) is a working title for what is to become a collection of ideas, practices, and tools for aiding and advancing the creative development of libre games (as in unrestricted in creativity by copyright or liability concerns) games. Manifest as in an ordered list (not as in manifesto, a declaration).

Other potential candidates pondered for a release title were - Libre Games Library (LGL) - Libre Games Laboratory (LGL)

We live in a world where economic concerns play a more or less part of our daily tasks and duties. Nevertheless, there are many that are willing or eager invest their spare time by engage their creativity on some task of their own satisfaction. One of those tasks can be game creation.

Why create games? Due to the narrative and artistic aspects added to technological development, game creation is as diverse as it can get. In game creation, there is room for many types of creativity, be it writing stories, scenes, or narratives, be it artistic content creation of all kinds, like images, textures, 3d models, animation, sound, music, cinematography, world, or character creation, through to the more structured creativity

of the kind of programming, coding, conceptualization, organization, management, public relation, or documentation. There is space for almost any type of creativity or interest.

Libre game creation not only challenges but also brings fun, utilizing your free time while benefiting all humanity, and can be a much more rewarding (if not even therapeutic) escape from the harsh reality of our life ('cause she is equally a harsh mistress). Creative engagement presents a much better reality escape route than e.g. television, web surfing, or drugs. It further presents a excellent playing field for personal growth.

Still, even in the age of global tech giants living off libre software without returns to original developers, we advocate the establishment of structures that will enable creators to fully exercise their creativity with as few obstacles as possible.

This project is an attempt at removing obstacles and reducing challenges found by contemporary and future libre game creators and developers.

## 1.1 Game Types

While we focus our attention on 3D words, the collected and established best practices may be useful to other game types.

We are a community of contributors to existing libre game projects and a forum for exchanging ideas and current and future practices in libre gaming.

We aggregate, collect, discuss but do not create games within this community. The latter is reserved to individual project communities and groups.

## 1.2 Game Vision

The originators envision a cinematographic exploratory free-style open-universe first-person adenture role playing game that is non-repetitive enticing and at the same time rewards the player for his spent time.

Think about a holodeckesque style of game as the guiding vision.

It should be a large cosmos possibilities and opportunities populated by star systems with spacefaring mechanics as well as possibility of roaming entire planets (or any other objects in space), in atmosphere, on the surface, on and sub-liquid, or below ground. Ships, buildings, structures, space stations, caves should be walkable and explorable.

Further specifics will be discussed in later chapters keeping in mind that an exemplary fantasy game may as well utilize a subset of the envisioned mechanics.

## 1.3 Open Source Games

Because (for those who care)

- we don't need repeated effort for the same type of games over and over and again

- for-benefit of many is more desirable than for-profit for few
- we must not be indirectly responsible for low-wage coding sweat shops
- we may be the only ones to boss oneself around
- creativity shall be free
- through creativity and safe play we learn to be gods in the overarching game play of life
- because we can

## 1.4 Open Source

Think twice, investigate, read or listen to opinions, if you expect to be making a living from open source. Though not impossible or unheard of, it requires a lot of commercial overhead.

Most of open source developers ate employed individuals who choose to spendvtheir free time doing somthing they consider useful while at the same time keeping their mind active and well excercised.

You must also be prepared for commercial organizations to sell and use your open code without any return to the original developers. This is the nature of open source. Be sure that you are prepared to walk this path.

## 1.5 Headsup Advice

- corporate strongarms will steal your work
- open source community has elements that do not care about responsibility or diligence
- the attrition and abandonement culture of some open projects can be nerve-straining
- some of the github project leaders' neglect of merging pull requests is outright disrespectful
- while open source should strive for unity of solutions, the previous realities lead to unsustainable fragmentation
- your body is the vehicle for your mind. the healthier the better your contributions
- equally, emotional health allows for better creativity and contributions
- there is naysayers, apes and savages in t-shirts abound, and they will be out to get you. just keep cool and detached.
- as all roads, this one is also full of sweat and disappointments (but also achievements and rewards)

## 1.6 About the Originator

Game industry is as old as humanity. Computer games have been around for well over 60 years (as of 25 Jan 2020) and the originator of this project (aka pyramid) has been creating and playing them for nearly 40 years.

Far from considering himself expert, he thinks of his contributions more as aggregatory with a slight visionary touch.

The aggregation of catalogued knowledge becomes more and more necessary with a very large knowledge base spread across the vast world wide web and more and more fractioning becoming prevalent. It should help lower the entry line for newcomers and make life easier for seasoned developers and creators.

On a side note: 1958's Tennis for Two is considered to be the first video game (source https://www.bnl.gov/about/history/firstvideo.php. For timeline of video games also see https://www.museumofplay.org/about/icheg/video-game-history/timeline

As early as 2017-08-30, pyramid started a collection of ideas for future games, albeit limited to a simple listing of objectives, features, and anti-features for games he would like to play, it was the spark that ignited this voyage.

## 1.7 Chapter Outline

The gaming challenges chapter is the manifesto of this document. It discusses desired game mechanics, technology, story telling, and other aspects relevant to creation and gameplay.

# 2 Gaming Challenges

## 2.1 Things to Consider

For our sci-fi roleplaying space opera game, we would like to see the following desired features (non-exhaustive)

- space travel
- walkable space ships
- docking to space objects
- planet flight and landings without cutscenes
- walkable planets without barriers
- terrain with progressive loading
- no loading screens for relocation
- loading screens only to private player dungeons that require teleporting (e.g. arenas)
- Unlimited skill-based experience levels
- Automatic idle progression (when not in game)
- Loot chests with rare items you need, otherwise useful mats
- personal growth aids with private ethical / unethical "kudos"

- non-combat sportive skill matches (e.g. races, knowledge competition, crafting competition)
- cooperative game modes (groups, parties, squads, operations)
- local and global chats

## 2.2 Things to Avoid

We would also strongly discourage the following anti-features:

- boredom from repetitive content
- coercion from grinding
- frustration from loot boxes
- losing items from death
- advancing only by killing pixels
- multiple currencies
- silent protagonist
- killing or destroying as main objective
- currency based economy
- oversized weapons
- limited skill treas (by trade or class)
- endless grinding
- in-app purchases
- in-app/world currency exchange
- random number cases to get your gear and keep you grinding
- unskippable cut-scenes
- MTX (micro-transaction) because the game devs don't respect your time, space, nor wallet.
- paid season passes
- bullshit DLC (DownLoadable Content)
- QTE (Quick-Time Events) You remember Dragon's Lair ? Yup, that's what modern "AAA" gaming has devolved into.
- grindfeasts aka Skinner boxes
- Flat UI that you can't fucking tell what are UI elements you can interact with vs static elements.
- Multiplayer games that don't allow you to run your own server

- "gaming industry" care for only how long they can keep "milking" you

## 2.3 Compelling Experience

Those compelling, enticing, reactive games with unique and realistic content that are able to engage the user for a long time will prevail.

Just like life, there is no knowing what will be. Games must become the same.

Usual objectives in casual gaming: - immersion (mixing your life with someone else's story) - pastime (actively doing something) - progression (sense of advancement and growth) - heroics (help humanity by eliminating monsters or saving a character from peril) - safety (fight the bad and evil and you will not really get hurt or die) - challenge (solve puzzles or problems to keep brain activity) - personal growth (learning collaboration, communication, ethics, bring out our better more cooperative instincts, discover positive evolution as a possible meaning of life, also learn functional knowledge through solving tasks) - social advancement (encourage evolution of cooperative behavior, learn to flourish and thrive together) - creativity (satisfaction from imagining worlds, characters, places, events and turning them virtual) - physical activity (limited, depends on technology like 3d controllers or holodeck)

There is little doubt that content development must swing towards artificially and intelligently procedurally generated universes in order to provide the games with enticing (immersive, challenging, safe, active) gaming experience.

## 2.4 Realism

I've said it before and I will probably say it many times again before I die. Most people have confused the concepts of realism with that of believability. They are NOT the same thing. And I think those people who keep saying "I don't want realism because it isn't fun" may not necessarily know it, but they have a more accurate idea what term means what.

**REALISM** is bringing real-world concepts and processes into the world being created (in this case the game world, but it could also refer to a movie environment or a book). There is one big reason why realism is difficult to pull off, especially in games. We are immersed in reality all the time, and we know what is realistic, because we see it every day. You can tell a bad magician or a poorly-Photoshopped image immediately because it just doesn't look right. In games it's even more difficult because we're still short on computing power needed to properly simulate a realistic environment.

**BELIEVABILITY** is much more forgiving, and most of the time it's a hell of a lot more fun to be a part of. Concepts of realism can be incorporated into a believable environment, and often are, but only to the extent that they add to the immersion. Anything beyond that is superfluous and is thrown away, and new rules are written. This makes believability much easier to achieve, and at the same time, much more difficult to pull off properly.

If a person is shot in an action movie, the realistic result is he drops to the floor. But that's not fun to watch, so instead the bullet knocks them off into the water. To enhance the beliveabiltiy, that happens every time someone gets shot (except for the main character, of course, he just bleeds). In our world, we need to get to a planet far away. Realism would say we need 200,000 years and a fuel tank the size of the moon, but that's not any fun. So we make up a magic super-fast travel system, and use it all the time. And we make up fictional documentation to back up our absurd creation.

We still know it's not realistic, but we believe it anyway. Why? Because we want to, and we can belive it "exactly the same" every time. The real secret to making a believeable environment is to take advantage of just enough of what reality does give us to help our immersion, then make up stuff to keep people interested, and use that made-up stuff in a way that is consistent throughout the entire game experience, offer "believable" (again NOT realistic) explanations in case people DO question it, and use this made-up material in such a way that the people playing WANT TO BELEIVE IT.

It's called **willing suspension of disbelief**, people. That's the secret. NOT realism.

There's another big strike against realism and games – real life is boring! people play games to get away from reality, not to become more immersed in it. If I want to have fun with reality, I'll go call my friends and we'll go outside and play football. author: pincushionman

And, to finish of the other side of that same thought, the key reason that people ask for realism (when looking for a consistent, immersive, and hence believable experience) is that, mundane as reality may often be, it is a consistent, immersive, and believable experience. Thus, what is more likely occurring (although there may be some who actually hunger for realism) is that, in expecting their experience to be like that of their normal reality, except in very particular ways, the discontinuities between the two sometimes become apparent, because of some artifact of modeling, or lack of experience with some particular phenomenon that is uncommon on dirtside. This then leaves them desiring some means for greater consistency, and they pull from the most consistent source they're familiar with.

Realism is useful - it's useful because anything we keep the same as what people expect it to be, is already consistent, immersive, and doesn't need to be explained. Reality, however, is just a starting point, like an archetypical makefile before being edited. author: jackS

source: https://forums.vega-strike.org/viewtopic.php?p=31090#p31090

## 2.5 What can be done procedurally

To avoid repetitiveness, content, in the long run, must be procedural

- space
- planets
- landscapes
- cities
- characters

- missions
- crafting recipes
- loot boxes
- gear
- stories
- vegetation
- wildlife
- ...

## 2.6 Content and Challenge Types

TODO: why this chapter?

- Story and mission based
- Skill and level based
- Creativity based
- Knowledge based

## 2.7 Design Documents

We separate the game content from the game play. The following documents are required for good game design

- Universe canon
- Game design

Universe canon describes history, events,species, culture, music, art, designs, vessels, personnae, galaxies, planets, places, fauna, flora, and everything that is needed to represent a believable universe.

The design document describes game mechanics, technologies, object standards, interfaces, libraries, coding standards, and everything required to make the game.

We **recommend** the structure for both documents derived from the one of the Vega Strike Universe Development Document and Game Design Document documents.

## 2.8 Documentation Standards and Formats

For the 2 types of documentation, canon and design appropriate libre standard formats shall be used.

It is required that formats support conversion and interchangeability (which most of libre formats do anyway).

We distinguish the following use cases: - shared scratch pads or forums may be used to develop a canon aspect, results should be ported to the master source document - master source document shall be used as the authoritative source of the canon - master source can be converted to various presentation formats (html, wiki, pdf, epub, ...)

An appropriate pipeline shall be established for the above documentation process.

**Recommendation**

- scratch pad: canon forum, e.g. https://forums.vega-strike.org/viewforum.php?f=28
- master document: Markdown (md), Open Document Text (odt), LaTex (tex)
- master document repository: game repository under doc/canon
- converted documents in repo directory: doc/canon/release

## 2.9 Open Fictional Universe Canons

There are a lot of interesting universes out there. Unfortunately, due to the state of general greed and copyright laws, it is not recommended to develop games based on published mainstream canon (think Star $$$$).

Public domain and libre licensed canons can be the basis for libre games, though care should be given to establish a viable consensus mechanism when enlarging the existing canon.

With procedural content, objects from the canon, e.g. places or characters, must be both, present in the universe, and not replaced by procedural content upon respawn.

Further care must be given in establishing authoritative bodies and decision processes in case canon needs to be furthered and enhanced.

**Recommendation** We reference here particularly the Vega Strike canon, principally devised by John Sampson aka jackS aka JS. Daniel Horn started Vega Strike in 1998 in high school. The engine and game are now orphaned but the canon is extensive and offers a good starting point for further development of our sci-fi game universe. The relevant document is the [vsudd][] "Vega Strike Universe Development Document".

For medieval type roleplaying games, WorlForge's Dural world, while not an as elaborate universe as Vega Strike, would be a good starting point.

See also:

- https://www.gamasutra.com/blogs/AlexanderFreed/20150615/246115/On_a_Lack_of_Origi

## 2.10 Story Writing

Engaging in non-repetitive story writing is a challenging undertaking. Many stories can be sampled from real life, art, and especially literature.

A word processor is your main tool for story writing. A chapter should be reserved for the manuscript with the big picture plot, personnages, notes, comments, and ideas.

Try Manuskript, the open-source tool for writers: http://www.theologeek.ch/manuskript/

See also: https://alternativeto.net/software/scrivener/?license=opensource, https://itsfoss.com/ope
source-tools-writers/, https://github.com/Blecki/TreeWriter, https://www.giuspen.com/cherrytree

Stories must be converted into workable code, usually using scripting.

See also:

- https://www.gamasutra.com/blogs/AlexanderFreed/20150629/247222/Six_Metrics_for_Bet
- https://www.gamasutra.com/blogs/AlexanderFreed/20150504/242101/Yes_You_Have_To_W

## 2.11 Storyboarding

Before moving on with asset production, the story can be optionally converted into a graphical sequence of events.

A libre standard storyboard exchange source format for stories is desired. Not only to make stories interchangeable but also to allow for future artificial algorithms to automatically add additional content to the game.

A storyboard exchange format (SXF) should at least contain informmation on

- personnages
- location
- time
- dependencies
- events
- triggers
- environment description
- scenes
- dialogues
- dialog options
- optionally images to visualize your story

Libraries for reading and writing storybords must exist, as well as user software for creators.

Explorable topics: screenwriting software, storyboard software, story writing software, storyboard templates

- https://github.com/wonderunit/storyboarder

## 2.12 Conversations

Conversations add immersion, realism, believability, engagement, and interactivity to the game play usually in one of the prevalent forms of branching conversations, with or without voice over, subtitles, or cinematographic scenes.

Branching conversations are a narrative tool for role-playing. They involve the player by giving him a sense of agency. They help establish the player character's personality and explore his emotions and inner life. Paired with voice over, subtitles, and cinematographic scenes they aid in convey a sense of believability or realism.

Instead of repeating ourselves, we will just summarize the important aspects of conversations to be considered when developing such and refer for further reading to the excellent article series Branching Conversation Systems by Alexander Freed at alexanderfreed.com.

Key concepts to consider

- Character emphasis
- Player character customization
- Branching Narratives
- Complex storylines with critical path
- Tool requirements
- Simple choice
- Hub and Spoke structure
- Waterfall structure
- Dialogue vs Narration
- Text manipulation (variables)
- Choice presentation (wheel, icons, paraphrases)
- Choice timing
- Forced player lines
- Marking critical path
- Skipping through conversations
- Merging "conversation" and "gameplay" games

***Conversation Tools of the trade***

- TreeLine Outliner Tool in Python
  http://treeline.bellz.org/
  Source code: https://github.com/doug-101/TreeLine
- Ink open source scripting markup
  https://www.github.com/inkle/ink Inky open source scripting editor: https://www.github.com/
- DlgSystem C++ Library and Tool
  https://gitlab.com/NotYetGames/DlgSystem/wikis/home
  https://gitlab.com/NotYetGames/DlgSystem
- ClearDialogue Java Dialogue Tool
  https://github.com/SkyAphid/ClearDialogue
  Example format: https://github.com/SkyAphid/ClearDialogue/blob/master/ClearDialogueAPI/
- Twine Tool for Interactive Stories
  http://twinery.org/
  Harlowe file format: https://twine2.neocities.org/
- Yarn Editor
  https://github.com/YarnSpinnerTool/YarnEditor
  https://yarnspinnertool.github.io/YarnEditor/
- Monologue Branching Dialogue Editor in Haxe (superseded by ClearDialogue)
  https://github.com/nospoone/monologue
  File format: https://github.com/nospoone/monologue.wiki.git

There is no recommendation as to the preferential tool yet, but a quick analysis highlights TreeLine or DlgSystem as potential candidates. TreeLine is a flexible and customizable tree node editor, but lacks a C++ library or game engine integration. DlgSystem comes with a library as well as authoring tools, allows for game engine integration, but seems

a bit more complicated to grasp.

**Conversation formats** must represent the conversation in such a way that it can be written, tested, debugged, and scripted. This means that in addition to text and the tree structure of the dialog object, it must include certain attributes that allow referencing audio, subtitles, and animation sequences. Further it should allow to contain information on which mood to apply when delivering voice-over and acting, what are the conditions triggering the conversation, variables that allow the dialogue to reflect previous player choices.

Such a standard format has yet to be established.

Recommended further reading:

- Branching Conversation Systems by Alexander Freed
- Defining Dialogue Systems by Brent Ellison

## 2.13 Game Mechanics

## 2.14 Player Character

## 2.15 Player Posessions

## 2.16 Character Progression

Instead of player levels, we recommend a per skill progression, advancing the particular skill, the more it is used. Slowly forgetting the skill during prolonged non-usage is an explorable option. Care must be given to balancing forgetting so as not to annoy and demotivate the player.

## 2.17 Autosaving

The autosaving feature immerses the player in the fictional world to a better extent. Care must be given that the player character is not locked into a situation without escape when reverting to a previous game state.

**Recommendation**

- auto save at predefined checkpoints
- create autosave slots at different intervals (equivalent to yearly, monthly, daily)

## 2.18 Time Progression

How fast should time progress in-game? How should the time progression be relative to the real out-of-game time?

In Vega Strike around 14h gameplay correspond to 1 Vega Strike Earth year.

## 2.19 Universal Time

In a space simulator game, there are different clocks on different planets, or in space, even different clocks across alien cultures. How would we establish a universal clock?

## 2.20 Game Engine

The burning question is: how to select an open source game engine? Some of the most popular open source game engines are listed under References Game Engines.

The most popular cinematic quality engine for cinema and games seems to be the Unreal Engine with just licensing terms and royalties depending on your project's income.

- Urho3D

We do not recommend:

- CryEngine though visually stunning with an impressive reference of major past games, it on ly supports non-libre WIndows based development.
- Unity though impressive and popular is closed-source and only free for personal use, thus not adequate to comunity projects.

Our **recommendation** goes to

- Godot Engine comes with innovative design and features, is extensible in C++ and scripting, script debugging, supports many major target platforms, and offers a wide range of authoring tools, as well as a friendly community, and much more.

## 2.21 Single or Multiplayer

Huge online universes can be fun through interactionn with other players though require many servers.

Thete is an inerent latency problem that may arise in universe instances with local servers and a gobal population.

Scaling of servers to users implies cost and time for maintenance.

## 2.22 Game Architecture

A very solid multiplayer architecture is presented in the World Forge project.

Cyphesis is the main WorldForge server. It provides everything needed in order to run a virtual world.

Mercator is primarily aimed at terrain for multiplayer online games. Mercator is designed in such a way that individual tiles can be generated on-the-fly from a very small source data set. Each tile uses a fast deterministic random number generation to ensure that identical results are produced "anytime, anywhere". This enables transmission of terrain

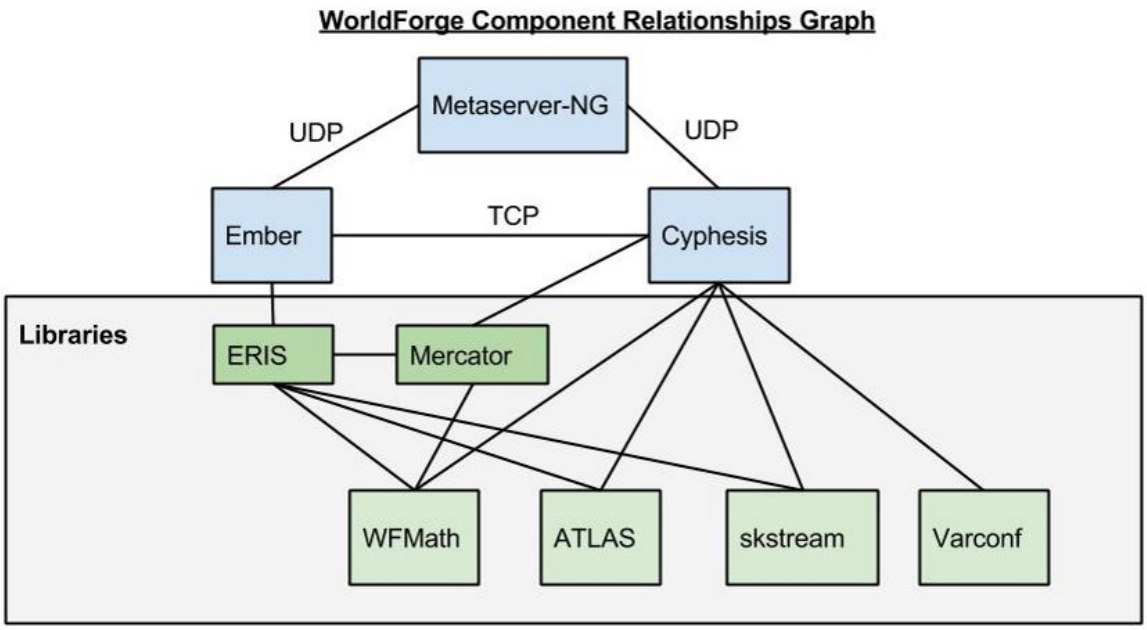


Figure 1: World Forge architecture

across low bandwidth links as part of the standard data stream, or server side collision detection with the same terrain that the player sees.

Atlas is the protocol which binds all of Worldforge together. It's a protocol meant to express a complete virtual worlds, and all communication between the servers and the clients uses it. The world itself as well as all actions that occur are all expressed through Atlas.

Eris is designed to simplify client development (and promote code reuse) by providing a common system to deal with the back-end Atlas tasks. Notably, Eris encapsulates most of the work in getting Atlas entities available on your client, and managing updates from the server. Thus it can be considered as a session layer above Atlas, providing persistent (for an entire gaming session) objects as opposed to transient Atlas ones.

## 2.23 Quests and Missions

The following concepts are current reference: - https://wiki.worldforge.org/wiki/Quest_Generation_
- https://github.com/vegastrike/VS-Design-Docs

## 2.24 Open Asset Formats

From experience, like in source code we must distinguish between

- source master formats
- exchange and transition formats
- release and distribution formats

Source formats are those that allow assets to be changed or remade. They are master formats because they allow for different exchange and release formats to be generated.

Transition formats allow for easy exchange between various asset tools.

Release formats must support the various requirements of an engine, most notably:

- describe smooth and edged geometry
- may support procedural geometry
- with procedural random release seed id
- describe texture vertices
- must store skeleton
- must store geometry animation
- describe geometry levels of detail
- contain or reference texture(s)
- indicate texture technology (PBR,...)
- must contain special non-renderable vertices, geometry, and vectors
- must contain model metadata
- contain or reference shader
- fast to load
- use least storage / memory (e.g. binary)
- may be non-human readable or editable
- must be debuggable
- must have a source code library for writing, reading, debugging
- be future extensible

**Recommendation**

- source: blender
- exchange: obj, collada
- release: engine specific

## 2.25 Serialization Formats

There are many formats available for serialization, the most commonly used being CSV, XML, JSON, YAML.

Our requiremnet for data serialization is that it would be:

- Editable with a text editor
- Human readable
- Machine readable
- Compact even for large data sets
- Libraries available in popular programming languages

Following the analysis and recommendations in this article, we equally **recommend JSON** as the select serialization data format for fulfilling all of the above criteria better than other formats.

[1] CSV vs XML vs JSON – Which is the Best Response Data Format?. Digital Hospital. 2016. Retrieved on 2020-02-23.

## 2.26 Games Within Games

Some say that “life is a game”. Actually it is a “play” and you are the player. Life is full of fakes: e.g. fake news, fake packaging, fake friends, and fake virtual reality games.

Within those games we are given the opportunity to be gods and create a better reality. Within those games we create minigames.

Games within games to chose in which one you want to grow.

## 2.27 Production Pipeline

From idea to finished release. All creativity starts with an idea. There is the player character, the non-player characters, the environment, and a a story to connect and relate them all and give the player something meaningful and entertaining to spend his time.

The story dictates which characters and environment assets will be designed. Visual Effects, voice, audio, and cut scenes may be required in additions.

## 2.28 Asset Pipeline

Will strongly depend on asset formats and types of games.

## 2.29 Asset Tools

A good practise when constructing new tools is to separte binaries for processing and visualization.

The asset pipeline should be automatable in the sense that bulk operations can be scripted.

By creating technology dependent user interfaces which call the command line procesd-ing tool we assure cross-platform as well as future portability.

## 2.30 Asset Catalogs

There is a variety of distributed asset management sites available today. Some of them offer free model.

Licensing must be considered as some of the free assets may be only for personal or academic use.

An aggregator service with search and filtering would enshorten the creators time when looking for appropriate libre assets for his creation.

## 2.31 Texture Rendering Systems

The traditional FFP (fixed function pipeline) shaders usually use this set of textures: - diffuse map - normal map - specular map - bump map

Ogre HLMS (High Level Material System) with PBS (Physically Based Shading) Ogre V2 support three basic types of workflow: Specular workflow, Specular as fresnel workflow and Metallic workflow with PBR (Physically-Based Rendering) textures.

Converting OGRE material files from FFP to HLMS is possible (see: http://www.ogre3d.org/tikiwiki/ti index.php?page=HLMS+Materials). For the conversion, the following tasks need to be accomplished.

1 Textures re-appropriation Required textures: diffuse, metallic, roughness. Optional textures: height, normal, ambient occlusion

Most existing FFP textures should be reusable from assets with diffuse, normal, and specular textures. For conversion we suggest this simplification: - PBR diffuse from FFP diffuse - PBR roughness from FFP specular - PBR metallic from 4x4 impostor black for most assets - PBR metallic from 4x4 impostor grey (intensity~235) for purely metallic assets (potentially reuse of FP specular) - PBR normal from FFP normal

2 Conversion of OGRE material description files Those files need to be converted (see: http://www.ogre3d.org/tikiwiki/tiki-index.php?page=HLMS+Materials) It should be possible to write a script that is able to convert the FFP .material format to the HLMS/PBS .material format.

3 Textures enhancement In order to improve rendering some textures might be need to be tweaked further - PBR diffuse intensity ranges need to be validated (min=30-50; max=240) - PBR diffuse reflectance for metals need to be validated (min=180; max=255) - PBR metallic for metals or mixed metal/dielectric assets need to be masked black for non-metallic parts - PBR metallic may also need to mask (black) metallic parts with artifacts - PBR metallic intensity range to be validated (min=235; max=255) - PBR roughness may be customized (where 0=smooth; 255=rough)

## 2.32 Terrain Tiling Issue

Graphical world representation uses meshes and textures to simulate visual properties of materials. Large terrains can become problematic when tiling (repeating) the same textures over a large terrain expanse. Due to distinctive features of the texture, they can become repetitive. Where texture tiling is very visible it will break the immersion for the player.

Terrain texture require

- sufficient variance and detail when close up
- non-repeating patterns when viewed from high up

Tiling is a recognized problem for environment artists. There is no unique cure, so tiling must be approached correctly from several fronts: texture preparation/generation, and texture rendering, props covering.

**Texture preparation** - Make textures have less distinct features. - Make texture details more uniform in hue and lightness

**Texture rendering** - Overlay other textures at different frequencies. - Add randomness to the texture, especially rotation, and size randomness. - Blend between multiple textures. This can be achieved through vertex blending or a mask texture tiling at a different frequencies, or a combination of both. - Add distance falloff blending - Add random noise layer

**Props covering** - Occlude far away terrain repetitions so the the tiling isn't as noticeable (prop coverage) with grass, trees, rocks, garbage, houses, or other objects.

**References**

[1] Improved Terrain Texture Tiling. larsbertram1. Retrieved on 2020-05-11. [2] Texture Repetition. Inigo Quilez. Retrieved on 2020-05-11. [3] Distance Blending. Freya Holmér. Last modified on 2014-03-16. Retrieved on 2020-05-11.

## 2.33 Planetary Textures

The image ratio horizontal:vertical must be 2:1 (assuming pixel ratio of the map is 1:1), since the texture is wrapped around the planet sphere horizontally around 360 degrees and vertically around 180 degrees. Necessarily, in order for the surface not to appear distorted, your pixel ratio of the generated texture must be 1.0, i.e. a circle must show as a circle when viewing the texture in an image viewer, on a monitor with square pixels.

The vertical and horizontal sizes should be a power of two (POT). Really, NPOT (non-power-of-two) textures are possible, but really, really, really troublesome. Don't use them. Just use POT. Love the POT. The POT is the mother, the POT is the father. Trust the POT.

Naturally, the images should be seamless (tilable) so that seams are not visible on the rotating planet, neither on the stitch nor at the poles.

3d rendering software with unwrap functions is recommended, since it is extremely inefficient and troublesome to create seamless textures in 2d programs.

source: text by pyramid from Vega Strike Development: Orbital Planet Surfaces

## 2.34 Procedural Content

- Procedural Content Generation

## 2.35 Procedural Textures

Here are some tools to review for making procedural game textures

- Material Maker
- Processing Framework
- Processing Source Code

## 2.36 Procedural Randomness

An interesting concept to consider is the random number generator used in Context Free Art procedural processor:

There is a nice system and user interface for managing the random seed and the scalable, distributed random number generators used to generate an image. This allows the user to reproduce an image. Steal it, please! (Credit appreciated, of course.) source: https://github.com/MtnViewJohn/context-free/wiki/About#any-code-worth-snagging

The random number seed is salted with an "entropy" value that is derived from the text of the cfdg file. Any change to the cfdg file will change the variations, even meaningless changes like changing '0.5' to '.5'. Adding or removing white space has no effect, so you can reformat the cfdg file as you wish.

Why do we do this? Context Free does not use a single, global random number generator. Instead each instance of a shape has its own random number seed. We use the venerable, but not so great, 48-bit linear congruent generator. Each shape instance passes a tweaked version of its seed to its child shape instances. This is all done so that you can render a given variation at a different resolution and get the same image, only with more detail.

Say you wanted to make a poster of a cfdg file. You would need a 10,000 x 10,000 pixel image. But when you are scanning for a good variation you want to render at, say, 500 x 500. With a global random number generator changing the resolution would completely change the variation. With local random number generators the variation looks about the same at all resolutions.

The 48-bit random number seed is salted with "entropy" because otherwise it doesn't work very well. The variations are all boring. We can't use a better random number generator because they all use lots of memory for state and we would need each shape instance to store its own state. And there can be millions of shapes. So we use the barely adequate 48-bit LCG algorithm and tweak it to keep it interesting.

Now in the next version of Context Free there will be global and local variables and you can change their value without affecting the "entropy". The entropy will be derived from the variable's name, not its value. There will be ways to animate designs using changing variables. source: https://www.contextfreeart.org/phpbb/viewtopic.php?f=4&t=895

Context Free uses a tree of random number seeds instead of a global pseudo-random number generator. This is done so that a variation doesn't change radically when you change the rendering resolution. To keep things interesting, the actual text of each shape replacement in a rule is reduced to a 64-bit value that we call its "entropy". This entropy value gets exclusive-ORed into the random seed during the shape replacement. If you change the text of a shape replacement then you change the entropy, which changes the randomness of the descendent shapes.

If you have a value that you want to change without affecting randomness then store this value in a global variable and reference the global variable in your shape rules. The entropy of a variable is derived from its name, not its value. So you can change a value

without affecting randomness.

references and sources (search: same seed): 1, 2

Godot engine implements the performant PCG 32 bit pseudorandom number generator.

As such, sufficient randomness and repeatability (across platforms) is assured and thus fulfils our needs for procedural generation.

Open topics to investigate next:

How do we assure repeatability of planet when refinement is dependent on camera position?

How do we store the seed hierarchy (e.g. system, planets and types, planet surface, biomes)?

Do we allow offline procedural preselection of seeds by the asset artist, to lets say select one of several procedurally generated biomes?

Do we generate on server (seed only) or on client (geometry) in a multi-player environment?

## 2.37 Procedural Universe Building

keywords: real-time procedural universe

listings - https://awesomeopensource.com/projects/procedural-generation

## 2.38 Procedural Starfields and Space Backgrounds

keywords

- procedural space backgrounds
- procedural starfields
- processing starfields
- generating starscape
- processing engine intergration

articles and code

- http://alexcpeterson.com/spacescape/
- https://petrocket.blogspot.com/
- https://github.com/petrocket/spacescape
- Shader based
- https://github.com/smcameron/cosmic-space-boxinator/tree/master/cosmicspaceboxinator
- http://www.mclelun.com/2015/10/blender3d-procedural-texture-starfield.html
- https://www.blendswap.com/blend/14059
- https://glitch.com/~starfield-maker-dev
- https://blenderartists.org/t/procedural-starfields-in-2-8/1160156
- https://github.com/yahiko00/Starfield

- https://www.overdraw.xyz/blog/2018/7/17/using-cellular-noise-to-generate-procedural-stars
- https://www.gamedev.net/forums/topic/690392-procedural-generated-2d-starfield-planet/
- https://www.shadertoy.com/view/XlfGRj
- Unity's ShaderLab: https://gist.github.com/CloudyWater/9dc32b60f73e4a3c300e067c11caa02
- https://github.com/slammayjammay/hyper-postprocessing/blob/master/examples/glsl/space-travel.glsl

## 2.39 Procedural Planet

keywords: open source seamless planetary flight, open source planet generator, Adaptive Mesh Refinement

articles

- http://vterrain.org/
- http://leah-lindner.com/blog/2016/10/10/planetrenderer_week1/
- http://leah-lindner.com/blog/et_engine/
- https://www.gamasutra.com/view/feature/131507/a_realtime_procedural_universe_.php?pr
- https://experilous.com/1/blog/post/procedural-planet-generation
- https://forum.unity.com/threads/realistic-openworld-workflow.593788/

source code

- https://github.com/Illation/ETEngine
- http://ratman.sourceforge.net/ contains P-BDAM algorithm
- http://www.firedrake.org/terraform/
- http://freshmeat.sourceforge.net/projects/exoflight/?branch_id=73789&release_id=274211
- https://github.com/sehugg/exoflight
- https://forum.unity.com/threads/roam-implementation-for-terrain-lod-c-help.70137/
- http://hhoppe.com/proj/geomclipmap/
- Pioneer
  https://github.com/pioneerspacesim/pioneer
- Pioneer Scout Plus
- GroundGrowing
- https://sourceforge.net/projects/wideland/
- Stylized Planet Generator for Godot
- Procedural Planets (Unity)
- Procedural Worlds for Unity
- Thalatta for Unity
- https://github.com/jpbetz/planet-generator
- https://zarkonnen.itch.io/planet-generator
- https://worldengine.readthedocs.io/en/latest/
- https://github.com/ZKasica/Planet-Generator
- https://www.findbestopensource.com/product/bauxitedev-stylized-planet-generator

- https://sourceforge.net/projects/fracplanet/
- https://sourceforge.net/projects/planetgenesis/
- https://sourceforge.net/projects/planetgenesis/
- https://forum.unity.com/threads/truly-procedural-planet-generator.198536/

catalogs

- http://vterrain.org/LOD/spherical.html
- http://vterrain.org/LOD/Papers/
- https://sourceforge.net/directory/graphics/graphics/3drendering/

## 2.40 Procedural Terrain

search: fractal terrain generation

articles - http://www.jgallant.com/procedurally-generating-wrapping-world-maps-in-unity-csharp-part-1/ - https://gamedevelopment.tutsplus.com/tutorials/unity-terrain-engine-tools–cms-28623 - https://www.shamusyoung.com/twentysidedtale/?p=202 - https://freegamer.blogspot.com/2009/03/open-source-3d-landscape-generators.html

sources - A Large Scale Analysis of Gradient Distribution in Procedural Terrain Generation - https://github.com/topics/procedural-terrain - WorldSynth - https://github.com/Scrawk/Interactive-Erosion - https://icecreamyou.github.io/THREE.Terrain/ - http://wwwtyro.github.io/procedural.js/planet1/

search: open source terrain

- https://alternativeto.net/software/nems-mega-3d-terrain-generator/?license=free
- https://code.google.com/archive/p/picogen/downloads
- https://forum.unity.com/threads/open-source-quixel-terrain-engine-smooth-voxel-terrain.240645/
- https://github.com/Chippington/Quixel
- http://paulbourke.net/geometry/polygonise/
- https://www.reddit.com/r/gamedev/comments/9pjeac/procedural_voxel_terrain_open_sou
- https://www.artifexterra.com/

## 2.41 Procedural World Building

listings

- http://vterrain.org/
- http://vterrain.org/Packages/NonCom/
- http://vterrain.org/Packages/Artificial/

## 2.42 Biosphere Building

## 2.43 Coordinate Systems

### Recommendation

We use the more common **right-handed cartesian coordinate system** (the same as in OpenGL). When using your right hand, your thumb points to the right (the +X axis), your index finger points up (the +Y axis), and your middle finger points towards you (the +Z axis). Note: a careful reevaluation should be done in around 10-20 years time should OpenGL be discontinued for the majority of game engines and replaced by Vulkan, since Vulkan's y-axis points downward.

For planet positions we use the counter-clockwise **spherical coordinate system** whereas the distance from origin is *rho*, the angle along the latitudonal equator in the xz-plane is *theta*, and the angular position along the longitude is *phi*. However, the azimuth *phi* is restricted to the interval (-180 deg, +180 deg), or (-*pi*, + *pi*) in radians. This is the standard convention for geographic longitude. Conversion between cartesian and polar coordinates must be performed whenever entering or leaving a planet and when landing on a planet.

For HUDs we use **2D cartesian coordinate system** with extensions between (0.0, 1.0). Whenever the screen aspect ratio differs from 1:1 the smaller screen extension will have an extension of (0.0+0.5/aspect_ration, 1.0-0.5/aspect_ratio).

```
// Example:
screen size = 1920 x 1080
aspect ratio / 1920 / 1080 = 1.78
x screen extension min, max = 0.0, 1.0
y screen extension min, max = (0.0+0.5/1.78, 1.0-0.5/1.78) = 0.28, 0.72
```

**References**


[1] Learn OpenGL - Coordinate Systems. Joey de Vries. Retrieved on 2020-02-23.

[2] Game Programming - Coordinates. WSI. 2013. Retrieved on 2020-02-23.

[3] 3D Graphics with OpenGL - Basic Theory. Chua Hock-Chuan. 2012. Retrieved on 2020-02-23.

[4] World in Motion - Part II. Positioning. Jason L. McKesson. 2012. Retrieved on 2020-02-23.

[5] Spherical coordinate system. Wikipedia. Last edited on 2020-02-18. Retrieved on 2020-02-23.


## 2.44 Unsorted Topics

What can we learn and which good practices can we carry forward from the original sandbox vision? What is the original sandbox vision?

Simple use of texture synthesis to transform one texture into a seamless tiling texture:

```
./texture-synthesis --threads 1 --inpaint masks/1_tile.jpg --out-size 1024 --tiling -o out/
./texture-synthesis --out out/out.png generate in/in.png
```

# 3 References

A collection of assorted references with the objective to shorten the entry barrier for new members.

## 3.1 Game Design Books

- Learn OpenGL
- The gamebook convention

## 3.2 Gaming Theory

- Open Gaming (https://en.wikipedia.org/wiki/Open_gaming)
- Open Game Systems (https://wiki.rpg.net/index.php/Open_Game_Systems)
- Circe Roleplaying System (http://worldforge.org/dev/content/rules/circe/)

## 3.3 Game Design Sites

- https://cgsociety.org/
- https://social.freegamedev.net/channel/devplanet
- https://social.freegamedev.net/channel/planet
- https://freegamer.blogspot.com/search/label/devcorner
- Gamasutra
- https://www.gamasutra.com/blogs/ChiragChopra/20190604/343845/Admiring_the_Game_
- https://gamedev.net/
- http://archive.gamedev.net/archive/index.html
- https://terranova.blogs.com/
- https://terranova.blogs.com/terra_nova/games/
- Physically Based Rendering: From Theory To Implementation

## 3.4 Conferencees

- http://jcgt.org/
- http://ianparberry.com/
- Gamedev.World
- SIGGRAPH
- SIGGRAPH Papers
- FOSDEM

## 3.5 Research Papers

- Lund University Computer Graphics

## 3.6 Technology Articles

- [Thinking in C++] (http://www.bruceeckel.com/ThinkingInCPP2e.html)
- [Adaptive planet and terrain mesh generation] (http://www.vterrain.org/LOD/Papers/index.htr
- [A Real-Time Procedural Universe] (http://sponeil.net/)
- [SFZ Engine Implementation] (http://sfz.schattenkind.net/wiki/index.php/Main_Page)
- Defining Dialogue Systems
- Dialogue Trees in Interactive Fiction

## 3.7 Game Developer Blogs

- Leah Lindner
- Alexander Freed
- https://erikhjortsberg.blogspot.com/
- http://dublin.alistairriddoch.org/
- https://kblin.blogspot.com/
- https://www.stevestreeting.com/
- Tynan Sylvester
- Alex Peterson
- Ryan Smith
- https://openteq.wordpress.com/portfolio/libregaming/
- Paul Bourke
- https://freegamer.blogspot.com/search/label/devcorner
- https://www.iquilezles.org/www/index.htm

## 3.8 Game Development Forums

- GameDev.net Forums
- IndieGamer Forums
- https://forums.cgsociety.org/
- TIG Forums
- IndieDB Forums
- Reddit Gamedev Topic
- Stack Exchange Game Development
- Kaidus Community
- Kongregate
- Dream.In.Code
- Buildbox Forum
- GitHub Community Forum
- Develteam
- Cocos Forums
- Unreal Engine Forums
- Unity Forum
- Python Game Development

- The Game Creators Developer Forums
- Nvidia Visual and Game Development Forums
- Ogre Forums
- https://forum.freegamedev.net/

## 3.9 Gamer Forums

- NeoGAF
- Gamespot Forums
- VideoGamer Forums
- PCGamer
- Gaming Latest
- GameFAQs
- Steam Forums
- IGN Boards
- Adventure Gamers - Forums
- MMORPG Forums
- The Verge Gaming Forums
- VGR
- AV Forums
- Eurogamer Forum
- Giant Bomb
- Escapist Forums
- Ars Technica Forums
- Game Revolution Forums
- Something Awful FOrums

## 3.10 Asset Artists Forums

- Blender Artists

## 3.11 Fictional Universes

- Science Fiction
  The Vega Strike Universe
  Vega Strike Universe Development Document
  https://github.com/vegastrike/VS-Universe-Lore-Docs
- Medieval Fantasy
  Dural
  Dural on World Forge Wiki
  https://wiki.worldforge.org/wiki/Dural

## 3.12 Story Writing

- Hemingway

- Dialog

## 3.13 2D Assets

e.g. portraits, background images

- http://opengameart.org/, also 3D, sound, music

## 3.14 Libre 3D Assets

- https://free3d.com/3d-models/vegastrike

May be free but not libre - Unreal Engine Free Assets - Unreal Engine Free Assets of the Month

## 3.15 Texture Assets

### 3.15.1 With Sources

Procedurally generated textures with source files that allow altering or reproduction of textures (e.g. with different resolution).

### 3.15.2 Release Only

Texture files in various image formats but without source files.

- https://www.deviantart.com
- http://texturelib.com

## 3.16 Shaders

## 3.17 Sounds

## 3.18 Music Assets

## 3.19 Game Engines

Listings - Wikipedia Listing of Game Engines

Popular engines

- Godot Engine
- Blender Game Engine
- Torque 3D
- CryEngine
- Urho3D
- Kha
- jMonkey Engine

- The Atomic Game Engine
- Castle Game Engine
- WorldForge

## 3.20 Game Libraries

- OGRE
- Armory Engine
- OpenMesh
- Collada
- Universal Scene Description
- WFMath geometric objects
- Varconf configuration files
- libWfut server/client asset synchronization
- Open Dynamics Engine
- OpenSubdiv
- OpenEXR
- OpenTimelineIO
- Ptex
- Computational Geometry Lib
- Godot Extended Libraries
- List of Modules for Godot engine
- A curated list of awesome C++ (or C) frameworks, libraries, resources
- Awesome list of C++ GameDev project
- Single-file public-domain/open source libraries with minimal dependencies

## 3.21 Graphic Content Creation Tools

- GIMP
- Listing from https://wiki.vega-strike.org/Links:Graphic_Applications
- AcademySoftwareFoundation Landscape
- ASWF Interactive Landscape
- OpenSourceVFX.org
- OpenFX
- OpenColorIO
- Open Shading Language
- Open Shading Language GitHub
- OpenVDB
- OpenBatchIO
- pfstools

- Alembic
- OpenFlipper
- SeExpr
- Physically Based Rendering: From Theory to Implementation Code
- LuxCoreRender PBR Engine

## 3.22 3D Modeling Applications

- Blender
- Cocos Creator
- Listing from OGRE Tools
- Listing from https://wiki.vega-strike.org/Links:3D_Applications

## 3.23 3D Texture Painting

- CinePainthttps://github.com/archont00/cinepaint-oyranos
- ArmorPaint

## 3.24 Software: Texture Processing

- Texture Synthesis

## 3.25 Procedural Planet Generators

- Fracplanet http://www.bottlenose.net/share/fracplanet/index.htm

https://gfindbestopensource.com/product/bauxitedev-stylized-planet-generator https://sourceforge.net/projects/fracplanet/ https://sourceforge.net/projects/planetgenesis/ https://sourceforge.net/projects/planetgenesis/ https://forumithub.com/jpbetz/planet-generator https://zarkonnen.itch.io/planet-generator https://worldengine.readthedocs.io/en/latest https://github.com/ZKasica/Planet-Generator https://www..unity.com/threads/truly-procedural-planet-generator.198536/

## 3.26 Terrain Generators

http://vterrain.org/Packages/NonCom/ http://vterrain.org/Packages/Artificial/ https://alternativeto.net/software/nems-mega-3d-terrain-generator/?license=free https://code.google.com/archive/p/picogen/downloads

https://forum.unity.com/threads/open-source-quixel-terrain-engine-smooth-voxel-terrain.240645/ https://github.com/Chippington/Quixel http://paulbourke.net/geometry/polygonis https://www.reddit.com/r/gamedev/comments/9pjeac/procedural_voxel_terrain_open_source/

https://gamedevelopment.tutsplus.com/tutorials/unity-terrain-engine-tools--cms-28623 https://www.shamusyoung.com/twentysidedtale/?p=202 https://freegamer.blogspot.com/2 source-3d-landscape-generators.html

https://www.artifexterra.com/ http://irrrpgbuilder.sourceforge.net/ http://lithosphere.codeflow.org https://bitbucket.org/gdecarpentier/scape/src/default/

https://picogen.org/ https://github.com/heremaps/tin-terrain https://cesiumjs.org/ https://github.com/heremaps/quantized-mesh-viewer

## 3.27 Godot Engine Tools and Plugins

articles - https://steincodes.tumblr.com/post/175407913859/introduction-to-procedural-generation-with-godot - https://medium.com/@swarnimarun/introduction-to-procedural-generation-with-godot-f24ea52532dc

tools - https://github.com/kidscancode/godot3_procgen_demos - https://github.com/Zylann/god

keywords: godot engine procedural generation

## 3.28 Audio Production

- Audacity

## 3.29 Movie Authoring

## 3.30 Procedural Quest Authoring

- Everquest Emulator
- Everquest Emulator Source

## 3.31 Notable Libre Games

- World Forge [https://www.worldforge.org/]
- Vega Strike [http://vegastrike.sourceforge.net/]
- Naev [http://naev.org/]
- PlaneShift [http://www.planeshift.it/]
- Oolite

Libre game listings - https://en.wikipedia.org/wiki/List_of_open-source_video_games - Listing from https://wiki.vega-strike.org/Links:Free_Games

## 3.32 Development Tools

- Easy Guide on using Git

## 3.33 Coding Standards

- Unreal Engine

## 3.34 Catalogs

- MagicTools
- GitHub Game Development
- PROCJAM Tutorials
- [https://www.gamedesigning.org/career/video-game-engines/]
- [https://www.gamedesigning.org/engines/bullet/]

# 4 Appendix - Libre Gaming Management Guidelines

## 4.1 Organization

As a global and intercultural community, we use internalional standards (e.g. SI units, ISO-8601 date and time formats within our discourse and in the products (games and tools) we provide.

Our dialog language is English (not restricted to any particular tomaytoe or tomaato). Localization is at the interested parties' leisure and not considered master or reference in any way.

## 4.2 Contribution

By submitting contributions the contributor agrees that his contribution will be included in this document under the license referred to in the License chapter.

## 4.3 Documentation

This present document is only one master document that collects and organizes various aspects of libre gaming practices on a high-level.

In-depth technical specifications are carrier by their own specification documents stored in the main Libre Gaming Manifest Project repository.

The main project repositories are listed on the project main page

- Libre Gaming Manifest Project (https://github.com/users/pyramid3d/projects/1)
- Libre Gaming Manifest (https://github.com/pyramid3d/libre-gaming-manifest) Libre Gaming Manifest documentation
- Libre Gaming Mirror (https://github.com/pyramid3d/libre-gaming-mirror) Libre Gaming reference documents mirrored

- Libre Gaming Assets (https://github.com/pyramid3d/libre-gaming-assets) Libre Gaming asset collection
- Libre Gaming Engines (https://github.com/pyramid3d/libre-gaming-engines) Libre Gaming engine related source code collections, libraries
- Libre Gaming Piepline (https://github.com/pyramid3d/libre-gaming-pipeline) Libre Gaming content pipeline related tool collections

The document can have multiple release instatiations in the form of various target release formats.

## 4.4 Consensus

A variety of opinion is encouraged. It drives innivation, creativity, and evolution.

Wheteever possible, multiple best practices are endorsed. Ideally, there is a delineation of the use case scenarios where practices for overlapping topics differ.

There are however certain decisions that are binary (not likely in our generic collection of practices, but in practical implementations thereof). Likewise, consensus to include or exclude some aspect into this documented collection might need to bd reached.

In such cases, we reach consensus by discoursive agreement, this means: 1 upon presentation of the idea or problem 2 we list out available options 3 describe adherents and deterrents 4 come to an informed agreement

Outdated techniques for consensus building are not suported and shall be altogether avoided: - authoritative decisions - representative voting - uninformed mob rule - attrition by trolling, ad-hominem, ... - meritocracy imposed consensus - group think guided

Silent agreement and decision by absence of philosophets are acceptable forms of agreement. Especially because agreements may be questioned and reopened again.

## 4.5 License



EOF