Libre Gaming Manifest

Present and Future

pyramid et al

release date: 2020-01-26

Contents

```
#
# @file   : libre-gaming-manifest.md
# @version: 2020-01-26
# @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 challanges 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 mistess). 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 excersise their creativity with as few obstacles as possible.

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

1.1 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.

1.2 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.

1.3 Game Vision

The originators envision of a game is an exploratory free-style open-universe first-person adenture style 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 a large cosmos of star systems with spacefaring mechanics as well as roaming the entire planet (or any other object 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.

1.4 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
- creativity shall be free
- through creativity and safe play we learn to be gods in the overarching game play of life

- because we can

1.5 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.

1.6 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)

1.7 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

1.8 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 ang gameplay.

2 Gaming Challenges

2.1 Things to avoid

- boredom from repetitive content
- coercion from grinding
- frustration from loot boxes
- losing items from death
- advancing only by killing pixels
- multiple currencies

2.2 Things to have

- Unlimited experience levels
- Automatic progression (when not in game too)
- Loot chests with rare items you need, otherwise usefull mats

2.3 Enticing Experience

Though game sales grow, those 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 (sence of advancement and growth) - safety (fight the bad and evil and you will not really get hurt or die) - challenge (solve puzzles or problems to keep brain activity) - heroics (help humanity by eliminating monsters or saving a character from peril) - 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 must be procedural - landscapes - cities - characters - missions - crafting recipes - loot boxes - gear - stories - wildlife

2.6 Content and Challenge Types

- 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 foloowing 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.

2.8 Open Fictional Universe Canon

While there are a lot of enticing 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.

Public domain and libre licensed canons van 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".

2.9 Game Architecture

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

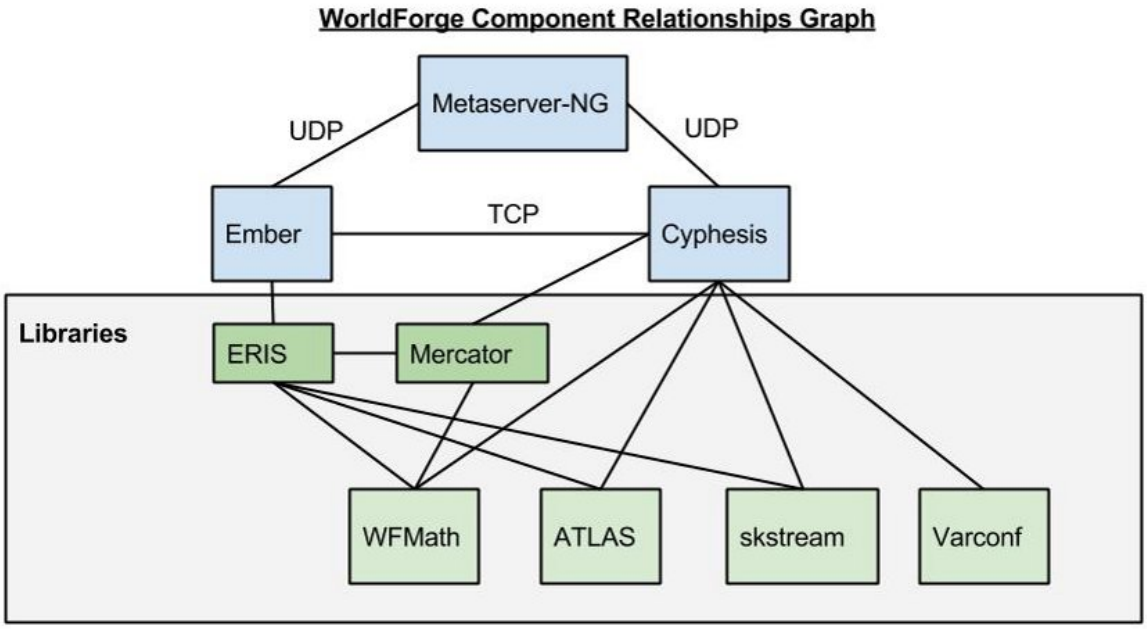

Figure 1: World Forge architecture

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 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.

2.10 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 - 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

2.11 Story Writing

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

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

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

2.12 Conversations

The following concepts are current reference: - https://wiki.worldforge.org/wiki/Conversation

2.13 Quests and Missions

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

2.14 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
- release: bxmf

2.15 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.

2.16 Asset Pipeline

Will strongly depend on asset formats and types of games.

2.17 Asset Tools

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

The asset pipeline shouls be automatable in the sense that bulk operations can becripted.

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

2.18 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.

2.19 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 reaproppriation 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)

2.20 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

2.21 Procedural Randomness

2.22 Universe Building

2.23 World Building

2.24 Biosphere Building

2.25 Game Mechanics

2.26 Game Engine

For our sci-fi game, we would like - space travel - walkable space ships - docking to space objects - planet 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)

2.27 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.

2.28 Player Character

2.29 Player Posessions

2.30 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.

2.31 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)

2.32 Time Progression

How should the time progression be relative to the real time.

2.33 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?

2.34 Unsorted Topics

What can we learn and which goo dpractices can we carry forward from the original sandbox vision? What is the original sandbox vision?

3 References

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

3.1 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/)

3.2 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
- 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/

3.3 Technology Articles

- Thinking in C++ http://www.bruceeckel.com/ThinkingInCPP2e.html
- Adaptive planet and terrain mesh generation http://www.vterrain.org/LOD/Papers/index.html
- A Real-Time Procedural Universe http://sponeil.net/

- SFZ Engine Implementation http://sfz.schattenkind.net/wiki/index.php/Main_Page

3.4 Game Developer Blogs

- https://erikhjortsberg.blogspot.com/
- http://dublin.alistairriddoch.org/
- https://kblin.blogspot.com/
- https://www.stevestreeting.com/

3.5 Game Development Forums

- https://forums.cgsociety.org/

3.6 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

3.7 2D Assets

e.g. portraits, background images

- http://opengameart.org/

3.8 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

3.9 Texture Assets

3.9.1 With Master Sources

3.9.2 Release Only

3.10 Shaders

3.11 Sounds

3.12 Music Assets

3.13 Game Engines

Popular engines

- Godot Engine
- jMonkey Engine
- The Atomic Game Engine
- Urho3D
- Blender Game Engine
- Kha
- Unreal Engine 4

Less known engines

- Castle Game Engine

3.14 Game Libraries

- OGRE
- Armory Engine
- WFMath geometric objects
- Varconf configuration files
- libWfut server/client asset synchronization

3.15 Graphic Content Creation Tools

- GIMP
- Listing from https://wiki.vega-strike.org/Links:Graphic_Applications

3.16 3D Modeling Applicationa

- Blender
- Cocos Creator
- Listing from OGRE Tools

- Listing from https://wiki.vega-strike.org/Links:3D_Applications

3.17 3D Texture Painting

- ArmorPaint

3.18 Procedural Modeling Tools

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

3.19 Audio Production

- Audacity

3.20 Movie Authoring

3.21 Story Visual Novel Authoring

3.22 Notable Libre Games

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

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

3.23 Development Tools

- Easy Guide on using Git

3.24 Coding Standards

- Unreal Engine

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 collections

- 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