The MonoGame Community. Wiki

Game Jam Manual [001]

Game Design Document
This part of the book covers the aspects a GDD covers.

You can use a physical notepad, a digital notepad file or a Word Document file, whatever you choose, be sure it is easily accessible wherever you are so that you can add to it as and when needed, a more advanced editor such as a Word Document, can give more flexibility as you can include images too.

Personally, I use Affintity Publisher.

Title: The name of the game

Whether you have a name already or not, always have at least a Project name.

Project X makes a good stand in.

A useful method to use is, maybe the Jam theme, if it does not work, a random title/project name generator could help too.

But having a name to use while developing can help you connect with the project on a deeper level and develop a focused interest to drive it forward.

Genre: The type of game, such as platformer, puzzle, shooter, etc. This helps focus the direction and mechanics of the project, as well as help you allocate a store category when distributing.

Combining genres is useful to capture a larger market, but being focussed on one genre can pay off too, be sure to list every aspect and then isolate the core aspects and use those words to describe the project.

Remember to start a document to keep track going forward.

Target Audience: The intended players of the game, such as casual, hardcore, children, adults, etc.

Say you were making a puzzle game, with anime characters, now the decision on if this was for grown ups or kids under typically 15 years of age, you may want to decide based on the style of the characters, also, hardcore can include very difficult to play games which can alienate your player base. Making casual games can pay off too!

Platform: The device or system that the game will run on, such as PC, mobile, console, web, etc.

It is crucial to understand the target platform, its input availability, its hardware specifics, capabilities, and distribution options; all of these can play a core role in your project setup. Typically a Game Jam is for PC distribution, if you find yourself being a Mac developer, consider installing Windows in a VM/second system so that you can test your app for both Windows and Linux.

Core Gameplay: The main mechanics and features of the game, such as movement, combat, puzzles, exploration, etc.

Understanding this aspect of your game, can help you specify a whole lot with the game engine development, the code structure, the elements you choose to develop, and the manner of connecting it all together.

Art Style: The visual aesthetic of the game, such as realistic, cartoon, pixel art, etc.

This aspect will decide the pipeline going forward, a 2D versus 3D pipeline carry vastly different processes and can carry time penalties on both sides if you are not aware of their individual peculiarities, and going for a specific aestetic by use of Shaders can have its own traps to trip you in your progress, be sure to keep things basic until you have a Minimum Viable Product MVP.

Sound and Music: The audio elements of the game, such as sound effects, background music, voice acting, etc.

Picture yourself standing in the middle of the city, close your eyes, what do you hear? A car passing by, the wind rustling the trees, people chatting on a bench in front of you, a dog barking at the youths, now inverse the senses, open your eyes and block your ears, how does it feel? Off right? This is what sound, and music, and voice; do for your game.

Find your rustling trees, your barking dogs, and passing cars.

Story and Characters: The narrative and personalities of the game, such as plot, setting, theme, protagonist, antagonist, etc.

How do you intend to drive the game forward for the player? Action? Choices? Puzzle Solved? Is there a storyline the player needs to follow and absorb? When or where is the game set? Earth? Space? 42BC? 1066? 2077? Whatever you choose, sticking to a basic variant can help you direct the art style, simplicity is not always easy, but worth aiming for should the time budget allow it.

Level Design: The structure and layout of the game world, such as maps, rooms, obstacles, enemies, items, etc.

It helps to create hand sketches of these and having them in front of you at all times for quick reference, it is no use if your protagonist enters a castle and is suddenly on a sandy beach! Items, and obstacles will also help you keep the environment in mind, a candle in a spaceship will feel awkward, but a torch in a dungeon will also feel out of place.
Let the elements dictate the environment, and direct the player.

User Interface: The elements that communicate information and feedback to the player, such as menus, buttons, icons, health bars, score counters, etc.

Hey, how did I get here? Oh that's right, I clicked the pause button on the screen, which clicked back at me, then I clicked Help, and here I am looking at how the game works. So, my ammo is doing ok, health needs some work, the backpack will help with that, and oh the menu has a slider to move up and down the content of the backpack. Be sure to keep menus simple and clear to read, images go a long way.

Example:

A game design document for a platformer game could look like this:

Title:
Super Jump Man

Genre:
Platformer

Target Audience:
Casual players of all ages

Platform:
Mobile

Core Gameplay:
The player controls a character who can jump and run across platforms and avoid enemies and hazards. The goal is to reach the end of each level without dying. The game has power-ups that give the character special abilities, such as double jump, invincibility, speed boost, etc.

Art Style:
Pixel art with bright colors and simple shapes

Sound and Music:
Retro-style chiptune music and sound effects

Story and Characters:
The character is a plumber who must rescue his girlfriend from an evil turtle king who kidnapped her. The game has humorous dialogue and cutscenes between levels.

Level Design:
The game has 10 levels with different themes and challenges. Each level has coins to collect and hidden secrets to find. Some levels have boss fights at the end.

User Interface:

The game has a simple touch screen interface with two buttons for jumping and running. The game shows the number of lives, coins and power-ups on the top of the screen.

Technical Game Document

This part of the book covers the aspects a TDD covers.

Engine:
The software tool or framework that the game is built with, such as Unity, Unreal Engine, MonoGame, etc.

Remember, what you put together is the engine, if something loads a file, that is part of the engine, if it draws to the screen, that is part of the engine, how you piece it together, is what results in the end product, and that is what you ship.

Programming Language:
The coding language that the game is written in or uses for scripting or logic, such as C#, C++, Python, JavaScript, etc.

Assets:

The resources that the game uses for its graphics, sound and music. These can be created by the developers or obtained from external sources. They should be compatible with the engine and platform.

Remember, to speed things up, grab asset packs, remember to replace them with custom assets before distributing, if this is for a game jam, you get kudos points for finding the time or a team to create custom assets, but remember, the immersion is the mechanics first, spend time on that.

Libraries and Plugins:
The additional software components that the game uses to add functionality or features that are not provided by the engine or language. These can be official or third-party. They should be compatible with the engine and platform.

Physics, Sound, Asset Loading. Usually a DLL, or an adjoined project, you can always craft your own added features, say a VR component to enable VR capabilities in your game, but for a game jam, this may be excessive work unless you have done it before, try to use ready to go kits in order to save time and effort.

Dependencies and Requirements: The software or hardware components that the game needs to run properly or optimally. These can include operating system versions, device specifications, network connections, storage space etc.

.Net frameworks to mention one, but anything you might need should be discoverable in the NUGET panel, try to keep on top of dependencies as they can completely destroy your workflow if you are unable to export them and link them up correctly, the fewer, the better.

Testing and Debugging:
The methods and tools that the developers use to check for errors or bugs in the game code or assets. These can include debuggers,

profilers,

loggers,

unit tests,

etc.

Remember to implement your own logging tools, a simple one is a text file that is written to using a new line entry method, and clearing the file for each fresh launch.

Distribution and Deployment: The methods and tools that the developers use to package and publish the game for its intended platform or audience.
These can include:

compilers,

exporters,

installers,

launchers,

etc.

Example:

A technical game document for a platformer game could look like this:

Engine:
MonoGame Custom Engine

Programming Language:
C#

Assets:

Graphics:
Pixel art sprites created with Aseprite

Sound and Music:
Chiptune music and sound effects created with BeepBox

Libraries and Plugins:

MonoGame.Extended:
A plugin for expanding the base MonoGame methods

MLEM:
A set of multipurpose libraries for MonoGame that provide abstractions, quality of life improvements and additional features

Dependencies and Requirements:

Operating System:
Android 4.4 or higher

Device Specifications:
Minimum 1 GB RAM and 150 MB storage space

Network Connection:
Not required

Testing and Debugging:

Debugger:
Visual Studio Debugging

Profiler:
DirectX Graphics Profiler

Logger:
Debug output text file

Unit Tests:
Visual Studio MSTest, Nunit, xUnit, etc.

Distribution and Deployment:

Compiler:
Visual Studio Build Compiler

Exporter:
Visual Studio Release Profile
Android Package

Installer:
APK file

Launcher:
Android device or emulator

Looking to participate in a Game Jam?

Want to make sure you start hitting the floor running?

Want to keep focused and driven?

Need a guidebook to help you with all of that?

Whether it is your first time, second time, or n^{th} time, this manual will help you!

Well, it will try.

Focusing on GDDs and TDDs, this manual will walk you through the process.

Use it or skip it, the choice is yours.

www.monogamecommunity.wiki