The Box Software information

The planning is for:
stages, interfaces, objects, tools, objectives.

The software engine have tools to do all kind of things, we can divide what it does in to stages. Often engines tend to do only one part, but the development is just one part, and often devided in sub parts. A software engine attempts to do all things.

Stages of the Software:

Design First stage,

design the project with the design tools, design document, game lore tool, add history and characters to the game using build concepts available.

For example why recreate the Orc concept, if there is already a very stable concept created.
This will make projects history much more consistent. If some one have the budget to create this things, ya, but if not is better to use something already done.

A good name could take many time to develop.

Project template Second stage,

Load a previous template with the basic concept on that theme we are using like, a rpg isometric game. We will already have a structure to work with, all files, maybe library connections, etc...

Libraries Third stage,

Have libraries to load a basic art to structure the project, for example a 3D low polly library. Which make all the game playable. For a initial game it does not need to have 100% art, this will also make possible to work on lower machines, test game and evolve.
I’m working in a 3D low poly librarie already set it up. For example put all kind of models which are very low in size.

Auto Code Fourth stage,

Add functions automatically, from code snippets or create own functions from standard functions tool without using the code editor. At this stage a person could have their hole project design in therms of code, just with a few clicks. It does not need to know programming to have the project working, or change some stuff in the templates.

Edit fifth stage,

design the basic stuff, like levels. Simple audio/video edit, interfaces. Small tasks that are needed to costumize your project. The inital stages can set a project with a few clicks, in here is necessary more work. This tools already exist, but if we develop some of them in the software it will auto make tasks, for example a code editor, which will link all things for us and have our code available.

Market Six stage,

Current market is expensive for Indies, and publicity, is very well protected who have some attention in the platform will put publicity.

The software will create a market so when you reach the stage to publish your project. Is available a platform for : founding, jobs, or help on the project.

Compile Seven stage,

Auto compile to platforms, width out the need for knowledge on how programming. Deal width platforms, free with out the need to pay more money for this things.

Publish Eight stage,

Was we see in the market stage it also create a free platform to publish. So if some one want to pay for your game, you will get all the value and not part of your work.

Divulge Nine stage,

Divulge the project with the social tool is like the market, have a costume platform which allow us to create the community,

Code

There is no problem in separeting things in to objects since all is global if we need lather is just a matter of moving things to other files and is still woring.

In this stage we will be doing a good experimentation, with objects.

Code Style

The Box uses a structure. All variables are global, and all files inherit, it’s a structure because all things are independent on one another.

The code is in file objects. The sub files inherit.

Instead of a image.h file that manage all we have a image.h with only the shared variables, and then a jpg.h that inherit directly from image.h

A Example

The library color.h :

The two named variable “color_text” is used because all variables are global, it prevent collision, or error in duplicate variables.

First name is also the object name “color.h”.

A use of color:

```
Include Color.h
color_text = "white";
color_background = "black";
use_colors();
```

function “use colors()” use variables declared before. They are not passed to functions but are global.

The sintax is a bit smiliar to standard C code, functions like getchar catch the information, and is not passed.

Project planning:

A Video editing, we may think is not needed another video editing software but if we do small video editing tasks in a project. A separate program may waste time. A program with the stuff already linked or with the options to link.

Incide the software we have access to: librarys, short cuts, automate tasks, auto links. If a program is in a software engine, is more productive then if it is outside. Have no access to information and can’t to automated tasks.

Structural Objects Files Planning

This files are theme for the structure of the application, all applications use this.

The files are in “application structures” folder. They define the structure of the application.

File.h

Converte libraries that have “file open” to this object. Libraries Like : images, sounds, 3D models. This will prevent to repeat variables like FILE and FOLDER. In all this libraries. This file object will be responsible to open files for all this libraries.

Or functionality to manage files, lets say our project is a management sistem we need to open files, we could use this library.

Image.h

The ideal is to have all images in it’s own librarie, for example jpg.h png.h If we only use jpgs in our project there is no need to include all images. This way we can choose what is needed.

Don’t know how much we can delegate in to image.h file object, but for what i read in images libraries, there are some common things used in all images. That could go in to the image object.

Variables.h

At this stage is not clear what variables object is for, but putting in there generic variables. Like counters which are declared in all function loops.

Is easy to move things up and down since all is global so it’nt a problema to separete things in to files.

Path.h

This is more or less used in other projects, the diferece we will be declaring a object directly just for this. It could deal with paths in diferent operating systems. Maybe some costume variables for the project.

Color.h

It will manage color, with generic variables. Is not very complex librarie, but there are many things that could use it like : textures, 3D, effects, text, interface, lights etc…

window.h

it manges our window, what is outputed in there. 2D, 3D, etc… at the moment there is a resolution.h but it probably it should go in here unless we are doing some kind of settings.

Control.h

In Graph App they implement a intermediary step in window,a draw area called control. So i’m thinking in also adding control, but in here we use more steps.

Like : window, control, draw, graphics, etc... If we have diferent methods for graphics like: 3D.h, 2D.h , Iso.h, holographic.h. We can separte like this:
window → control→ “type of grahics “ -> draw → interface → font → mouse, keyboard

Draw.h

Since C is so comlex, that it uses functions to draw to window, think it will be better to have it’s on object, so it is more separeted. We could have diferent kinds of draw types.

Think mention before, it may have or not diferent types, since all is global it does not really matter what kind of experiences we do with the code. Because changing something it will not affect as much as nested code.

layer.h

Manage layers for diferent kind of things, like 2D editing, 3D, images, textures, colors, interfaces, etc…

Don’t know if control could use it since it start to layer things.

Interface.h

It deals with all kind of stuff needed to manage interfaces. It deals only with the graphic part, the text is in font.h .

for now the object is : property_menu, property_button;

3D.h

Thinking in putting all the basic stuff needed for the 3D, like coordinates, the models, textures, etc... will not need do re declare the basic stuff.

Not sure what is common in this 3D didn’t study this part, but all that can delegate should be in here.

Only the coords x, y, z are used in diferent kind of things.

Texture.h

Texture is connected with the 3D, but don’t know it could be reused, like in a editing texture program. At first it does not seem we could reause, since is something connected with 3D, but we lather see we may need to reuse.

As mention since all is global is just a matter of moving the code from files. No problem in over or less separating.

2D.h

This could be image.h since, is more or less the same thing, generic information for images or could be a generic object to work with games, like layers, but in that case is better to delegate to interface.h.

Is a bit like 3D don’t now that much about it. Maybe some kind of effects could go in here, they seem that could be shared like 2D editing effects and game 2D effects.

Iso.h, holographic.h.

If we are writting costume Control and window we should be able to use graphics for this types. As refered 2D, 3D, ISO, Holograhic

The concept is that isometic.h should have stuff that is proper for this style. Don’t know much about this, but if does not have that much could go in to image.h

Arithemetic.h

Also called the “math” librarie in some cases, it will have the math for all things, don’t know if is realy needed at this stage, it could be in files like 3D.h, so they have their specific math for them.

Probably is something not that much sharable. If can’t share there is no need to create a new object just for that. But could be used in other projects, that need a math library.

.font.h

A part of the interface is delegated in to this object, it manage all parts of text.
Like, style, type and management of fonts.

Type.h

Is something not need in high languages, but in C I see it scattered throw the projects. It is necessary to deal width diferent types of operating system.

We have our types of data in a shareble librarie object.

Memory.h

The allocation of memory is also something very scattered throw the projects , think in creating a object for this. Most things allocate memory, so is only logical that we have our functions in this.

At this stage don't know how this works. But from the planning perspective is correct to have file object for this.

For what i know of C at the moment, buffer, seem thing that could go in here.

Don't know if is possible to writte a generic memory to work with several types of things. Depend a bit on the type of working, that's why people use a lot pointers.

.string.h

Strings in C are very complex, but don't know if this is not stuff for the variables.h which will manage our variables.

costume manage strings.

mouse.h / keys.h

We could have diferent type of operating sistems, generic management for this things.

Structure File Objects

Work the same way but are specific for tools for this project.

.tool.h

This is a specific librarie for the box, since it abstract diferent kind of tools to be used throw our tools.(Abstraction can also be called sharable things.)

In here we are sharing tools information that is use in 2 or more tools.

It have the index keys, to store information.

Function.h

Automate functions, by creating a way of recoring functions information. In a way is a object of the object.

Sintax : funtion_name; funtion_file; funtion_language;

It could also help a bit in debug since, the program have to get this information in real time, if we put it there it speed this process.

Game Structures File Objects

They are in “Game structures” folder. To structure the game.

The objects to deal with functionality found in games.

At this stage game objects are not that much develop since they first need is to have the application structure or the box structure working so, is more a speculation at this stage.

It was in first version, but still not that much develop.

Money.h

A specific object to deal with money in games.

We coul go with a evolution.h object and put all aspects of evolution in this object: money, xp, progress, etc… But in the project mapper, we couldn’t add project with actions.

Let’s say we want create a automate project and we want : money, npcs, xp, enviorment. We click on them, it inclues money.h, npcs.h, xp.h then it create the file project this items will be added. As simple was knowing their dependcys. If is in evolution.h will be including more that it need.

Is something I didn’t do, but it needs to be done.

The library money,h -> depends on : game, window, control, etc…

If we know library dependency’s then we could create auto projects.

Class.h

Another feature that is used in many games, although is still not declared in a standard librarie is the Class.

The “Game lore tool” add lore to game, pre build information, which will be talked next. Will be a matter of connection. One could create the class of the character in the game lore tool and it passes to the code as a programming class.

Debug tools

A folder that have Costume programs to debug individual parts of the program. For example, images, fonts, interface, etc…

Since this is not woring with the framework we are developing the framewor. In a framework you develop the code and can test.

In here we can’t test that easy, think the best option is to develop and test individual parts and them mount the working version.

The files are in the folder “debug_tools” . All the stuff that is develop to test individual parts will be in there.

It output all the informamation of what we are doing like :

the type if is 3D model; all the information inside our model; all the information connected with the diferent parts of loading;

Not only debug but could also work to test files; software debug and files debug.

This help because error maybe in the file and not in the software this way is easy to detect where the error is comming from.

Information storage

This files are in the “information” folder.

We are doing diferent kind of actions, with files, so we need a efficiented way to store information.

First think in adding XML like files, but then since i know bit of databases create a simplified database using files.

The user do not need to read the file. Why put so many infomation in there? When the software can load and translate that information for the user. Means that the files will be much smaller, allow for larger amount of information stored.

Also thinking in adding index to files, so we can have 2 files, one of indexes and other with the information. To search the file one just need to search the indexes which have much less informtion, then go in the file where the line is in the information file. Think will be much faster.

Index keys example ; id ; name ; discription ; files;

file example : id 1, the box, software, the box/

A index file : line 10, id 1, the box; line 20, id 2, game dev

Works like a database but with just text files.

The tool.h have the index keys to use the information folder.

The object have the key index to know what represants what, so its many of information not stored in the file, because we will be repeating : Title, discrition, etc…

keys indexs : project id, project name, project discription, etc…

File : key 1 is project id, key 2 project name, key 3 project discription .

OS structures :

Graph App, uses programs outside the software, to deal with several operating systems. Is something i'm still studying, Don't know how will be done in this software. For now adopting Graph App style. Added the folder for the several operating systems.

Tutorials in files

Added tutorials to the top of programming files , which allow to easy follow the information, and learn C.

Reading a manual to start doing a simple tasks can be very consuming of time.

In this methog there is no need to read the hole manual, the information is divided in parts, just need to read individual parts to follow.

Tutorial tool example

- "C language, files used : main.c, tool.h, variable.h, path.h, "

Link the information to the task we are doing, avoid searching files, where are the tutorials. to improve the access to information.

The software uses the principle of reduce redundancy, not only in code, but in development tasks. A person should have enough capability to produce a project by it self.