FreeYourSoul Online

Release 0.0.1

Quentin Balland

Aug 12, 2022

CONTENTS

CHAPTER

ONE

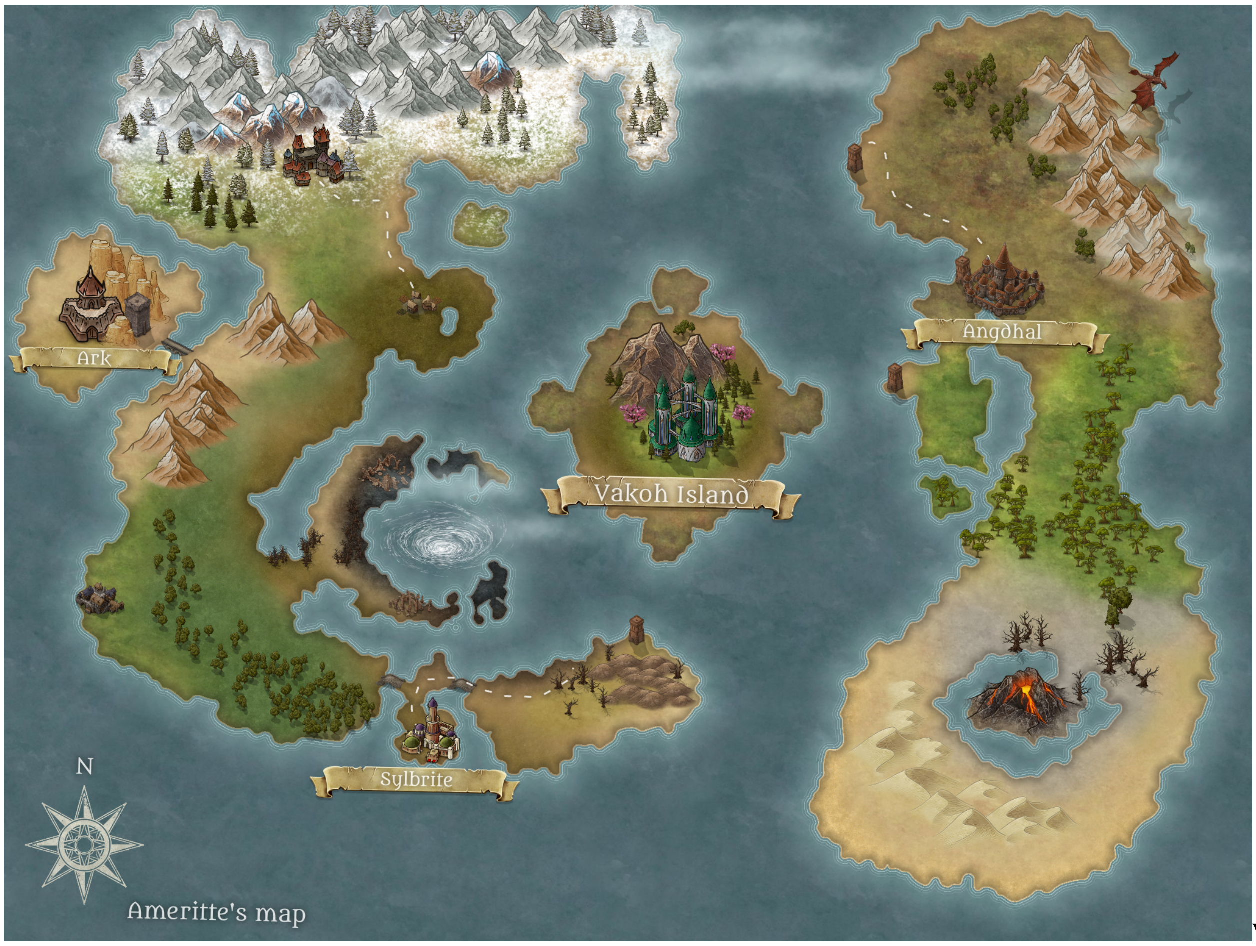
FREEYOURSOUL

1.1 Factions:

- Renegades
- Guardians
- Empire

1.2 Introduction

Welcome to Amerite



WorldMap

Long time ago, goddess was walking along humans, living on the Vakoh, a floating island with a worshipping people, the Guardians while the other humans stayed living below. Granting them favor for their fidelity, they wanted to keep this people by their side. To do so, they shackled their soul into their body in order for them to not be able to die and called them Guardians of Vakoh, with the hope of the guardians worshipping them forever. But the goddesses didn't bother ask the permission to do such thing, which irritated some of their followers, whom centuries after centuries living, felt the feeling of wanting more. The Goddesses accepted granting a favor to them, and decided to land Vakoh, the holy island, in the great sea, in order to let the Guardians live aside with the other normal humans. Centuries passed, and the irritation of some Guardians towards their immortality progressed into hatred. Living side by side with their humans peer made them realize how lonely they were. Seeing loved one leave while they felt trapped on earth. They started calling their divine gift a curse. They asked the goddess to release the soul bounding spell. The Goddess refused as they grew attached to all of their followers. The fidel Guardians didn't approve and forgive such a disrespectful request and decided to exile them, and renamed them Renegades.Having to quit the holy island and their goddess protection, they decided to continue living with the humans.

A pact is made, the Guardians returned in the holy island and shared some of their knowledge about souls given by the Goddess in order to quench the greed of the humans. The Renegades taken as responsible of the war and departure of the Goddess, didn't have the right to go back to Vakoh and had to live further in the mountains, still looking for a way to release themselves from their own bodies.

1.3 Improvements [14/06/2020] : Andrea & Quentin > Online Meeting

1.3.1 Game play

- having tactical view and on tactical and classical view.
  - Having a little screen above that show the owned characters and highlight the ones that has to take their turns.
  - Having the order list of action (in tactical of the selected side, in the classical of the current side)
  - in tactical : Having a listing view of all friendly characters (to be able to select them to go in the classical view of their side or just get health/mp information for healers)
- Having a mapping between hexagon sides and in-world sides. (make it possible for eccentric setup).
- Ambush is a state in which you enter a fight and the enemies can hit you from an adjacent side. (in term of UI, during an attack an ambushers appear the time of the attack and disappear)
- Encounter wise (trails in the sky like system)
- Having a log of actions only in classical view (would be a mess in tactical)
- Being able to play the turn of a character wherever he is in any view mode (if target required having a popup-like window to select target).

1.3.2 Story wise

- The Renegades are the same (wants to become mortal). But the initial start of their rebellion was caused by the Goddesses deciding to send off SOME immortals to a a "greater place". But some of them stayed for centuries without having the ascension. Making them rebel against the goddess, making them leave. Which made the flying island of Vakoh fall off from the sky to the Ocean. And that is why they got excused by the immortals to become renegades.
- The renegades, not being able to be ascended anymore, wants to find death to go to the "greater" place humans claims to go when they die.
- The Immortals are extremist (mostly) that wants to convert people to their religious thought in order to make the Goddesses come back and discover ascension (which is impossible now that the goddess left).
- Humans follow their greed to get the secret of immortality by experimenting on souls and improving their science.
- Focus on the maelstrom as ariana line for the story.
- Reminder : The maelstrom happened at the closest place human can reach by foot to the Vakoh island. It's the place the biggest garrison of the human army was standing. Making their number, after the incident, decrease too much to continue to fight. Signing the peace.
- Maelstrom has been created because of a "failed" human experimentation (maybe wrong ?) which impacted how the souls interact in Amerite, making them physically manipulable by anyone. Strangely the monsters/creatures got angrier and more aggressive following this event.
- Maelstrom is actually a gate to a parallel plane that is feeding itself on Amerite.
- The Governor of Sylbrite is possessed by **an entity** from that world, making him looking like a good guy fighting for the peace between races and factions.
- The Governor of Sylbrite is manipulating the world in order to enter in a war (getting more souls to feed to his homeland)

- The story of each factions starts (following a story telling in each capitals) by a tournament organised by the Governor of Sylbrite having for main official purpose to celebrate the 50 years of peace after the great war. Actually the factions takes it as an opportunity to show off the power that they still have in case of a re-start of hostility. The governor is using it to create tensions (TODO idea still has to be dug on) between the factions.
- End Game idea : The war cannot be stopped in time, but still a fight occurs to dethrone the Governor (en fight). But still, because mmorpg never dies, the two world merge making a new land to explore and more actual enemies conflicting with Amerite.
- For foreshadowing purpose : humans Souls, when they die, are actually sent over the other plane in order to be reincarnated. And the "Ascension" of the immortals is actually a manual sending made by the goddess to go into this other plane (funny enough the Renegades where right to want to die). TODO Idea still has to be dug on
- The renegades doesn't hold a grudge against the Immortals as they don't care about being a noble in Vakoh but just want to free themselves from earthly shackles.
- The chief of the renegades is a famous warrior (the most ancient and powerful immortals) that stopped the bloody battle of Rigs with his power alone. Making him a symbol of peace and power to the citizen of any race in the world.
- The researcher is the reason why humans and renegades can't be all friendly (torture and in-human experimentation made to find out the secret of immortality).
- The researcher "The einstein of Amerite" is the cause of the maelstrom incident, it was not an accident as he was trying to prove his theory of another plane was true. The experiment actually was a success.
- Idea This guy could be the entity that possessed the Governor, TODO need to find a reason why.
- Idea : Maybe having the immortals being a perfect democracy, where every voice count (and actually count) and the respect between each other (in the same species) is absolute. Making it the "Teddy Bear" of the world (so the target), it is a closed country, having strong resent toward human realm. But a party (the church one) is defending from the outside the most they can.

CHAPTER TWO

CODE DOCUMENTATION!

Code documentation can be found in Doxygen here : https://codedocs.xyz/FreeYourSoul/FyS

CHAPTER THREE

PROTOCOL

3.1 Client & WorldServer

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

3.2 Client & ArenaServer

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

3.3 Client & InventoryServer

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

```
// FlatBuffer
```

3.4 Internals

3.4.1 WorldServer & ArenaServer

```
// FlatBuffer
```

```
// FlatBuffer
```

3.4.2 ArenaServer & InventoryServer

```
// FlatBuffer
```

```
// FlatBuffer
```

CHAPTER

FOUR

ARENA SERVICE DOCUMENTATION!

4.1 Turn per Turn priority

4.2 Requirements

- A full turn is defined as follow: The slowest pit participant played sign the completion of the current turn
- A player's character (or a contender) can have multiple action in a single turn depending to its speed
- The priority list has to be recalculated when :
  - A player's character (or contender) die and is temporarily (for player's character) or definitely removed from the priority list.
  - When a player or a contender join the fight (as for character reviving)
  - The speed of a character has been impacted (by a spell, an item, an attack).

4.3 Implementation

Each pit participant has a speed defined as an unsigned integer. The faster go before the slowest.

Generation of priorityList :

1. Order the list of speed from fastest to slowest.
2. The first one is the first one to have a turn (add him in the priority list as first)
3. Subtract his speed with the second one
4. Re-order the list of speed from fastest to slowest.
5. The first one in this list is added in priority list
6. Subtract his speed with the second one
7. Re-order..
8. …

Graphical example, each color represent a different character (monster or player's character):

turn1

The difference of speed between the fastest and the slowest being important, the purple player can play 4 times before the end of the turn.For the second turn, the base maximum speed is added to the every one plus their own base speed.

turn2

This turn is way shorter than the previous turn (every character play only once), but the next one as the speed stabilized a little bit, but in the third turn, as the purple player end this turn with 19 of speed, he will play multiple times before the end of the turn.This algorithm give an edge to the fast player without being broken (playing multiple time every single turns).

Here a resume of the two turns:

resume

4.4 Arena

The Arena is the name for the service that is managing FyS Online fighting phase. An instance of a fight is called a **Fighting Pit**.In further explanation, the service will be called Arena and Fighting Pit will be called FP (for faster writing, because as any developer, I am lazy to write down long names).

4.4.1 Arena specification

The arena will have multiple requirement to fulfill:

- First it will be required for the Arena to be a service able to host multiple Fighting Pits (FP).
- Any Arena instance can host a fight from any player (no character restriction depending on its position for example).
- A fight hosted by one Arena will be managed from begin to end on the same arena (which will make the service stateful).

4.5 Fighting Pit

The fighting pit (FP) is an instance of a fight in the world of FyS Online. FyS Online has a turn per turn battle system called SCTB (**S**ide **C**onditional **T**imed **B**attle).This is strongly inspired from the Final Fantasy X battle (CTB : Conditional Turn Based Battle) but is quite different in pratice.

4.5.1 Reminder of how CTB (FFX battle system)

Exanation taken from finalfantasy.fandom.com

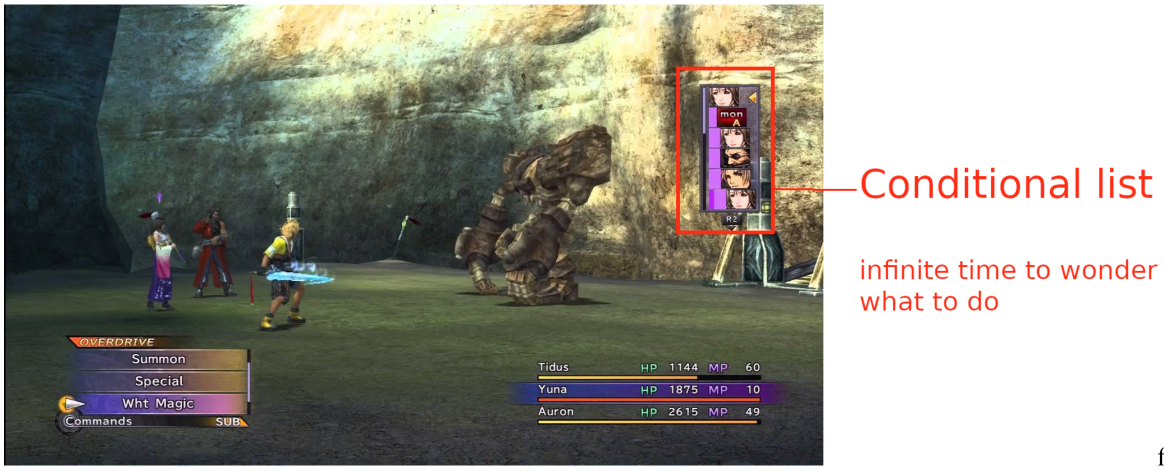
CTB is a turn-based system which does not operate in rounds, instead it uses an Act List that is affected through various means and thus does not guarantee that each participant in a battle will have an equal number of turns. Units with higher speed take more turns than slower ones, making speed more important than in other turn-based battle systems. Players can substitute party members mid-battle adding a new level of strategy.

Resume

It is a fighting system based on a ordering list of the player based on the speed/statistic of each player and monsters. Which means that a fast character will have the right to attack multiple times before a slow monster fight back.

4.5.2 Why making a new battle system?

First because its funnier to invent an original thing instead of copying existing concept. On top of that, while I loved FFX, I think some issue in the gameplay makes it a little bit too easy to play. Final Fantasy X is not particularly famous for being hard. And I try to solve those issue while also adding some elements making the gameplay maybe more enjoyable in a MMORPG context (multiplayer).I would like first to thanks CarNage64 whom is a twitch stream that speedrun FF7. He gave me advices on how the FFX is imperfect and how it could be improved.



ffx-fight

What are the downsides of CTB?

The way CTB is designed present several issues:

- Infinite time turn: Turns are potentially infinite, everything is based on the priority list displayed on the UI which is calculated based on the speed of your characters (and enemies). Which gives you all the time you want to elaborate a strategy, to heal your hurt characters at the last second (knowing perfectly that the opponent won't attack before you heal). This last point being important as the "stress of potential death" is almost not present. This point is certainly the biggest flaw in CTB.
- Too easy to change characters: In CTB, you have 4 characters fighting and many more are waiting behind and can replace your character currently in play. This is a nice thing as it increase the number of playable character making the game more interesting. But the issue is that there is no penalty changing character in-fight (the turn of the switched character not being lost). This also makes the game easier as you can retreat your hurt character before the enemy attack, and makes him come back afterwards basically for free.
- Not adapted for multiplayer

4.6 A new battle system SCTB (Side Condition Timed Battle)

Side Condition Timed Battle is a mix between CTB (FF-X battle system) and more classical battle system based on timers.Let's peel the SCTB acronyme in order to understand the basics of this battle system.

4.6.1 Condition Timed

Even if FF-X condition list, which make you able to know the ordering of the characters/opponnents turn, made the game easy. It was a pretty good idea as it gave an interesting strategic aspect to the battle, making you able to better play with the "limit" of your characters ressources (nearly dying, nearly out of magic power and so on...).SCTB has the same condition list (based on characters/monsters speed), but in order to remove the easiness of FF-X battle system, each characters turn are timed. Plus a configurable difficulty mode makes you able to change this timer to be shorter (easy) or longer (hard). At the end of the timer, the character turn is skipped and the next character in the condition list takes it place.

This "Condition Timed" part of SCTB remove the Infinite time turn problematic explained above.

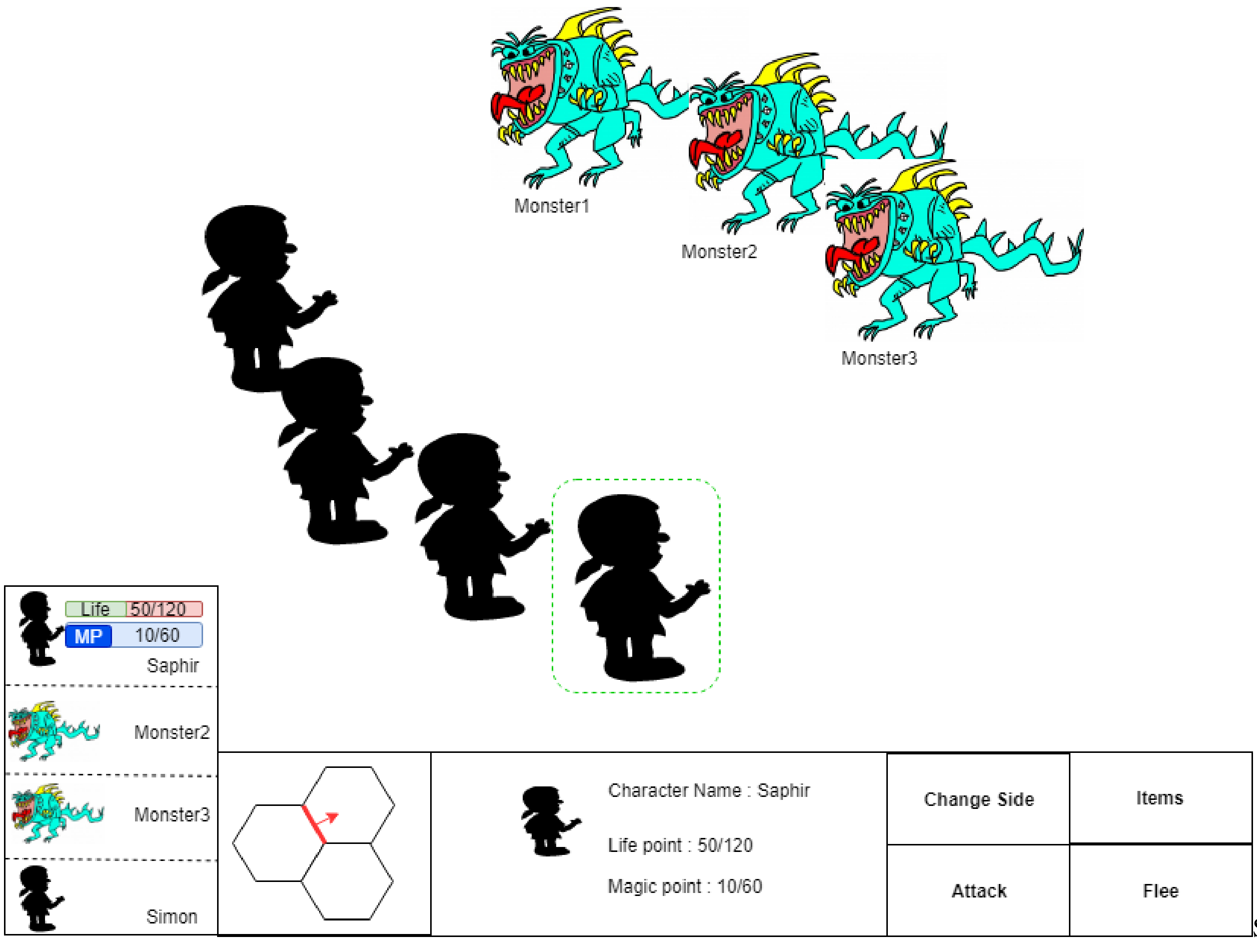
4.6.2 Side

Here is where SCTB makes a difference compared to other turn per turn battle system.The field in which you fight your opponnent(s) is not flat like in pokemon (or Final Fantasy 1 to 10). But is composed of a total of 18 possible angles. The gaming board could be represented as 3 hexagons stuck together (see ascii art below).Your team of characters (4 characters) will be on a SIDE of the below hexagon, same for the monster(s). Your characters can move independently from one side to another (adjacent side) to fight monsters.

```
                      _____
                     /     \
               _____/  B    \        18 possible playable sides
              /     \       /        6  possible ambush sides [A_NE, B_SW, A_SE, C_NW,
↪ C_N, B_S]
             /  A    \_____/
             \       /     \
              \_____/  C    \
                    \       /
                     \_____/
```

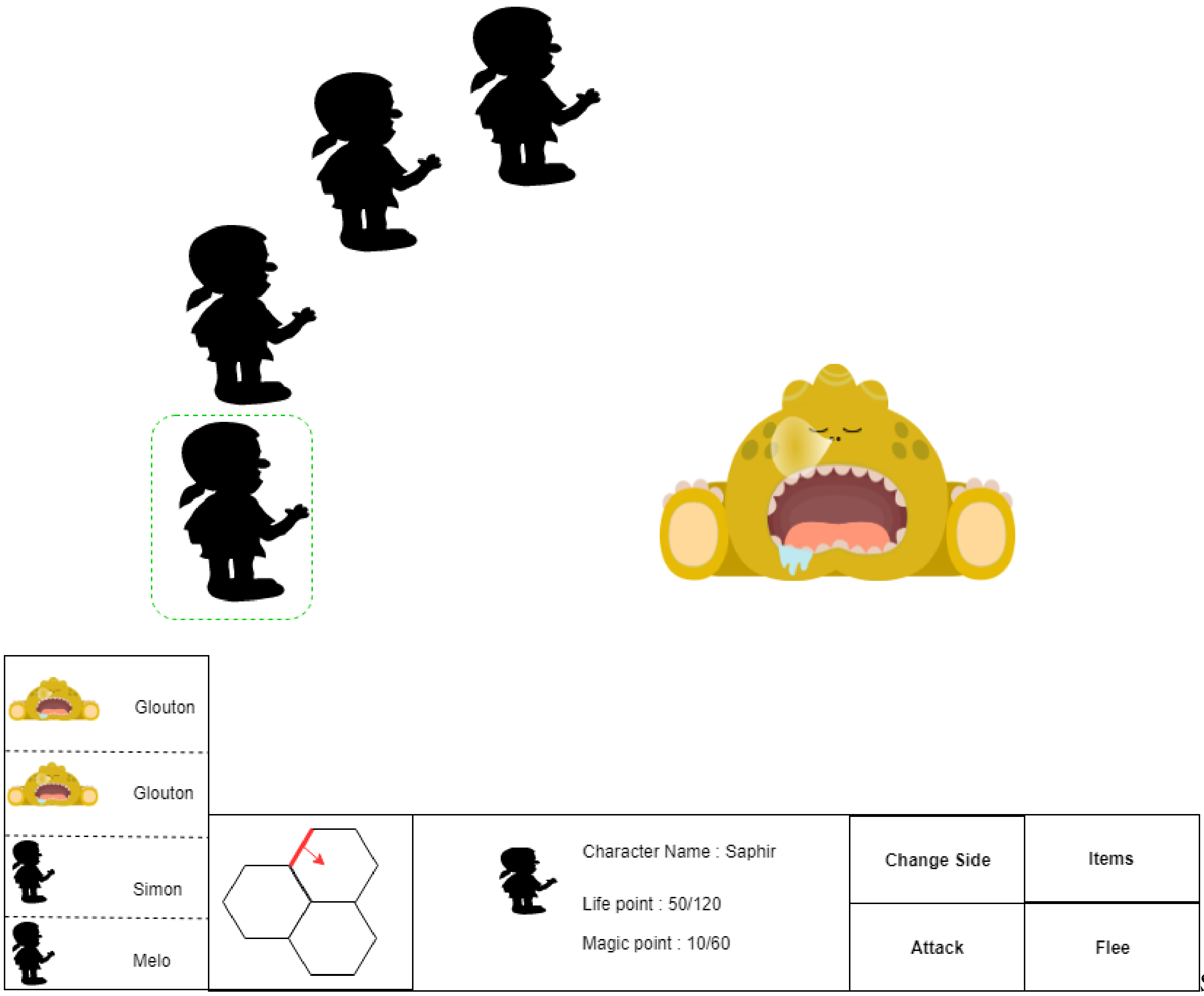
Little monster (the size of your character) are going to be able to be on only one side of one of the three hexagon (the same as you character). Which means that if a character change sides, if the monster doesn't follow you, he won't be able to reach this monster from the new side.You can thing of each sides as a separate battle happenning at the same time. There can be 18 battle front in total, and your characters can use a turn to change battle front (side) instead of doing another action (attacking for example).This aspect add some strategy as some monster could have very powerfull area of effect spells that would touch ever.

You can find bellow a very sketchy draw of example:



SCTB-MoveExample

In this state the player is in the at the south west of the hexagon B, if the player decide to move all his characters to the left, the characters will be on the North west of Polygon B (as shown below). In this other side of the polygon B, another monster will be present.



SCTB-MoveExampleAfter

Large sized monster

A Monsters can be big enough to be attacked from multiple sides. For example a big Hydra Boss monster could be on the 6 faces of the Hexagon A at the same time. Some characters of your team would be on the border north-east of the Hexagon A to fight one of the Head, others could be on the north border of the hexagon to fight another head. Or even at the back of the hydra (on the border south / south-east) where the characters would fight the tail of the Hydra.

Multiplayer adapted

18 battle front possible is really a lot, way more than all the normal encounters that will be usually made in the game (1 to (rarely) 3 battle front in random encounter). So why having that many of them ? The answer is simple, in MMORPG, two things are really appreciated by the players, the PvP (player vs player) to show off the individual skill of each players. And the PvE (Player vs Environment) to play in team and make succesful strategy against the game's traps. FyS Online is focusing on PvE for now, and the SCTB is particularly interesting for it as it would be the first (as far as I know) turn per turn game that would allow multiple player to play at the same time in a very good pace (not having to wait long before playing).

Multiple player that has to wait their turn? : Yeah, sounds horrible right? if there is a maximum of 18 battle field with, let's say 2 player by battle field. Then there is 36 players. You should wait for 36 player + the monsters turn before playing. It is impossible. But, what if each side, each battle field, have their own turns? Then it means for each battle field (each sides of the hexagon), you just have to wait for 1 player and the monster present on this specific side to play before your turn. And this is how SCTB works, there is one timer and one priority list by side, which makes it playable even with a lot of players playing at the same time.The fact that you can move from one side to another is not inconsequential as most of the spells you can cast are limited to the current side your character is in, so strategy can be put in place in order to have healers coming to heal and protect themselves on a side that is less dangerous (for example).Of course if every players are present on the same side, it will have the waiting issue. But it shouldn't be the case as the only reason to have lot of player fighting together is to beat a boss that will be a "larged sized monster" as explained above. Whom will have multiple sides to fight one and adds (monster that could spawn during the different phase of the boss), also on other sides.

But you said each player can move their character from side to side independently: Yes, making the characters able to move independently is the goal. It would increase the complexity of the game exponentially for the players (as they will need to handle multiple turn at the same time from different sides). But it can have the issue of making too many players on one specific sides and thus slowing down the pace of the game greatly. Additional work on this part is required to make the gameplay fun.

4.7 Integrate Chaiscript

The goal of this documentation page is to explain how to integrate new scripts (Contender and/or Actions) in ArenaService and make it usable to the players.

4.7.1 Introduction : Chaiscript

ChaiScript is a scripting language made by Jason Turner designed to be used with C++. Its main focus is in its usability and interfacing with C++. While keeping good performances (less than LUA but still acceptable) it makes it possible to do an almost transparent interface between C++ and Chai, making it a very suitable scripting language for complicated game logic.

Existing C++/Chai Mapping

Before explaining how to make new script, it is important to note that helper functions have been registered and exposed to the ChaiScript engine to give access to information for the script. Those ChaiScript function are registered in the file ChaiRegister.cpp.

Below are the C++ signatures of those functions, they are accessible and callable from chai scripts:

```
/**
 * Generate a random number between the double range_InBound and range_OutBound
 * @return a randomly generated number (double)
 */
double generateRandomNumber(double range_InBound, double range_OutBound);
//! Overload with int
int generateRandomNumber(int range_InBound, int range_OutBound);

/**
 * @param isContender true if the Character to check is a contender, false if it is an␣
↪Ally
 * @param id id of the Character to check
 * @param orient Side to check
 * @return true if the character is on the given side, false otherwise
```

(continues on next page)

(continued from previous page)

```
 */
bool isCharacterOnSide(bool isContender, unsigned id, HexagonSide::Orientation orient);

/**
 * Check if a character is on an adjacent side (adjacent side beeing a side you can move␣
↪to)
 *
 * @param isContender true if the Character to check is a contender, false if it is an␣
↪Ally
 * @param id id of the Character to check
 * @param orient Side to check
 * @return true if the character is adjacent to the given side, false otherwise
 */
bool isCharacterOnAdjacentSide(bool isContender, unsigned id, HexagonSide::Orientation␣
↪side);

/**
 * @param lhs a side to check
 * @param rhs a side to check
 * @return true if the two given sides are adjacent, false otherwise
 */
bool isSideAdjacentSide(HexagonSide::Orientation lhs, HexagonSide::Orientation rhs);

/**
 * @param alterations alterations to add
 * @param replaceIfExist true if the alteration override existing ones, false otherwise
 * @return
 */
void addOnTurnAlterations(data::Status& status, std::vector<data::Alteration>␣
↪alterations, bool replaceIfExist);
//! Same function, but the alteration will be applied before the turn occurs
void addBeforeTurnAlterations(data::Status& status, std::vector<data::Alteration>␣
↪alterations, bool replaceIfExist);
//! Same function, but the alteration will be applied after the turn occurs
void addAfterTurnAlterations(data::Status& status, std::vector<data::Alteration>␣
↪alterations, bool replaceIfExist);
```

On top of the above functions registered reference to types present in the C++ worlds exist in order to access to the status of the characters

```
//! name of the reference to use in Chai scripts of the instance of PitContenders
pitContenders;

//==== PitContenders chai exposed member methods ====

/**
 * @param comparator against which the contenders will be checked against
 * @return the contender following the comparator the most
 */
FightingContender PitContenders::selectSuitableContender(HexagonSide::Orientation side);
//! Same but retrieve an alive character
FightingContender PitContenders::selectSuitableContenderAlive();
```

(continues on next page)

(continued from previous page)

```
//! Same but on a specific side
FightingContender PitContenders::selectSuitableContenderOnSide(HexagonSide::Orientation␣
↪side, ComparatorSelection<FightingContender> comp);
//! Same but only retrieve an alive character on a specific side
FightingContender PitContenders::selectSuitableContenderOnSideAlive();

//! Retrieve a randomly chosen alive character on a specific side
FightingContender␣
↪PitContenders::selectRandomContenderOnSideAlive(HexagonSide::Orientation side,␣
↪ComparatorSelection<FightingContender> comp);

//! Retrieve a contender with a specific id
FightingContender PitContenders::getFightingContender(unsigned id);

//! Retrieve all the contender on a specific side
vector<FightingContender> PitContenders::getContenderOnSide(HexagonSide::Orientation␣
↪side);

/////////////////////////////////////////////////////////////////////////////////////
↪/////////////////

//! name of the reference to use in Chai scripts of the instance of AllyPartyTeams
allyPartyTeams;

//==== AllyPartyTeams chai exposed member methods ====
/**
 * @param comp against which the allies will be checked against
 * @return the ally following the comparator the most
 */
TeamMember AllyPartyTeams::selectSuitableMember(ComparatorSelection<TeamMember> comp);
//! same but retrieve an alive character
TeamMember AllyPartyTeams::selectSuitableMemberAlive(ComparatorSelection<TeamMember>␣
↪comp);
//! same but retrieve a character on a specific side
TeamMember AllyPartyTeams::selectSuitableMemberOnSide(HexagonSide::Orientation side,␣
↪ComparatorSelection<TeamMember> comp);
//! same but retrieve an alive character on a specific side
TeamMember AllyPartyTeams::selectSuitableMemberOnSideAlive(HexagonSide::Orientation side,
↪ ComparatorSelection<TeamMember> comp);

//! Retrieve a randomly chosen alive character on a specific side
TeamMember AllyPartyTeams::selectRandomMemberOnSideAlive();
```

4.7.2 Integrate AI scripts

Template

Chaiscript for AI are represented by a class which has to take a contenderId and a level as parameter in constructor and follow this implementation:

| to follow | descriptions | | |————————- | ——————————————————————————————————————
| | Constructor(id, level) | Constructor taking the level of the contender and its id, this constructor has to set the actions attribute | | runScriptedAction(id) | which will determine and apply an action which will be following the next one | | setupContender() | which will set the spawning position | | contains actions | The contender class has an action attributes, which will be a map of Action chai object |

```
class Sampy {

    attr id;
    attr level;
    attr actions;

    def Sampy(contenderId, level) {
        this.id = contenderId;
        this.level = level;
        this.actions = action(

            // possibles actions/decide target vector
            [
                action(ACTION_SCRIPT(50), fun(action, thisContender) {
                    // decision strategy
                }, "test:key:1"),

                action(ACTION_SCRIPT(80), fun(action, thisContender) {
                    // decision strategy
                }, "test:key:2"),

                action(ACTION_SCRIPT(0), fun(action, thisContender) {
                    // decision strategy
                }, "test:key:3")
            ],

            // decide target function
            fun(currentContenderStatus) {
                // return index of the action to do
            }, ""
        );
    }

    def setupContender() {
        var &thisContender = pitContenders.getFightingContender(this.id);
        var &thisStatus = thisContender.accessStatus();
        // Set the status of this contender
     }

    def runScriptedAction(id) {
        var &thisContender = pitContenders.getFightingContender(id);
```

(continues on next page)

(continued from previous page)

```
        var &thisStatus = thisContender.accessStatus();
        var actionId = this.actions.decisionStrategy(thisStatus);
        var &action = this.actions.act[actionId];
        return action.act.execute(action.decisionStrategy(action.act, thisContender));
    }
```

Action attribute

You noticed above the attribute action, which is a core part of the artificial intelligence as it is the place where the decision making is done concerning the target, and which spell is going to be cast.

```
class action {
    attr act;
    attr decisionStrategy;
    attr actionKey;

    def action(a, decisionStrategy, actionKey)
    {
        this.set_explicit(true);
        this.act = a;
        this.decisionStrategy = decisionStrategy;
        this.actionKey = actionKey;
    }

}
```

The action object is a simple chai object defined as above, containing an attack object (see below how to create such chaiscript object), a lambda function that return the status of the target for the given attack, and the key of the attack.

This action attribute is firstly set with a vector of action and a function to decide which action to do. This is the decide action part. And each action object in this vector contains an actual action and a function that select a target for it.

```
    def runScriptedAction(id) {
        var &thisContender = pitContenders.getFightingContender(id);
        // retrieve status of the current contender
        var &thisStatus = thisContender.accessStatus();

        // decide action to do thanks to the action attribute containing the vector of␣
↪actions. We call the decisionStrategy that decide and return the index in the vecotr␣
↪of the action to do.
        var actionId = this.actions.decisionStrategy(thisStatus);
        // retrieve the action object thanks to this index
        var &action = this.actions.act[actionId];
        // execute this action on the target retrieved thanks to the decisionStrategy of␣
↪the action
        return action.act.execute(action.decisionStrategy(action.act, thisContender));
    }
```

4.7.3 Integrate attacks scripts

Template

```
```

4.7.4 Integrate alterations script

Template

4.7.5 Include actions

CHAPTER

FIVE

CACHE MANAGER LIBRARY (CML)

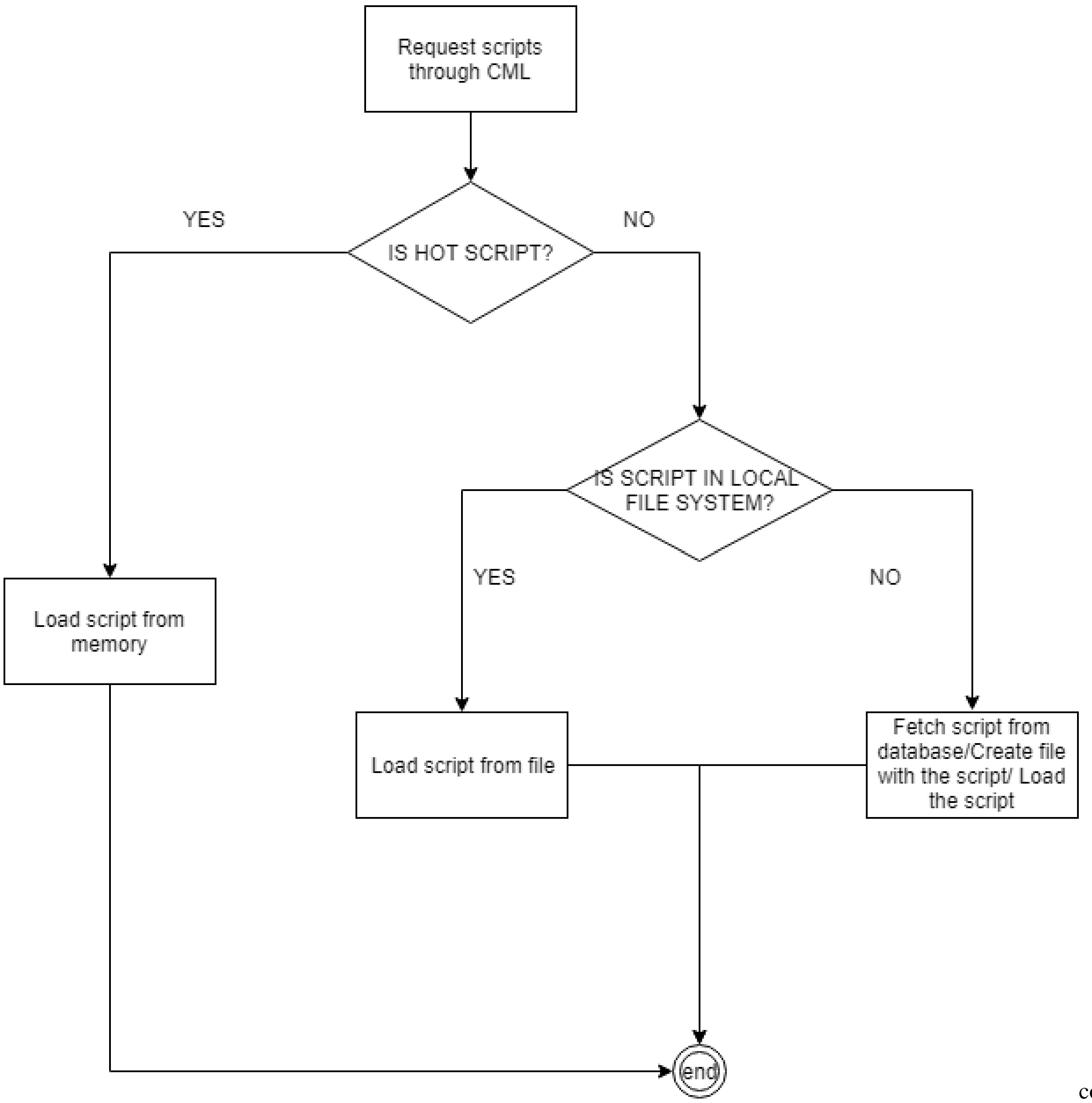
This is a library that is used in order to retrieve data from the database, and cache it in order to not have expensive db call.

For example;in the case of the Arena Service (Battle service). At the start of a battle, the contenders (monsters) that the player are facing are selected and thus known. The Service will have to fetch how the contenders behaviour are (stats, attacks, artificial intelligence and so on..), this logic is stored as chai scripts in the DB. When the battle end, if necessary, the memory is flushed for the next battle (new contenders / actions to fetch).But in case the same contender is selected for the next fight, it would be a waster to retrieve the behaviours twice in the database.

5.1 To resolve this issue : CML is composed of 3 distincts steps:

- Retrieve from the database
- Create a at a specified file system location a folder hierarchy in order to find the contender again if needed.
- If the scripts is considered as "hot" (often used), it is kept in memory and not flushed at the end of the battle.

Thanks to those 3 steps, in the second attempt to retrieve the contender data. 1. First it will be checked if it is a hot scripts (basically contenders are not hot scripts, but actions like attacks are). 2. If it is not a hot script, the local file system is checking for the existance of the file that should correspond the script.3. If it is found, it will be loaded, otherwise the database is requested.


Request scripts through CML
IS HOT SCRIPT?
YES
NO
Load script from memory
IS SCRIPT IN LOCAL FILE SYSTEM?
YES
NO
Load script from file
Fetch script from database/Create file with the script/ Load the script
end

conditionalDiagram

5.2 Up to date

In order to not have not up to date scripts stored in the file system. Two mecanisms are possible:

- Cleanup of the script at back-end startup (to enforce new database fetch)
- Redownload from database of the scripts stored on the file system at the back-end startup (to ensure up to date data).

CHAPTER

SIX

WHAT IS A SERVICE?

CHAPTER

SEVEN

WHAT ARE THE DISPATCHERS?

CHAPTER

EIGHT

INDICES AND TABLES

- search