# Our project short description

Fire Emblam game was inspired by well-known game series Fire Emblem. It features incredible combat experience. It uses complex combat system PekosYs. Each player can create his own team of champions in fancy character creation menu. Then teams fight on nice game map. Game also features great character images. You will see three windows. It the center there is main game map, on the left side there is event log that shows all events like dealing damage etc. On the right there is side menu that shows all information about selected champion. Due to the big problems with Pharo (accidental crashes) and Iceberg there isn’t implemented as much features as we wanted.

# Instalation

1. Download and install Bloc-core

*Metacello new*

*baseline: 'Bloc';*

*repository: 'github://pharo-graphics/Bloc:pharo6.1/src';*

*load: #core*

1. Pull from gitlab (<https://gitlab.fit.cvut.cz/sedlam34/BI-OOP_FireEmblemRPG.git>)
2. Then in Playground type: „fireEmblam start.“ to start the new game.

# Basic architecture of our solution

Here is description of our tags in the package FireEmblamRPG:

## FileEmblamRPG

Here are our champion classes. This tag is highly test covered. There is Polymorphism in Champion class, each class is subclass of Champion class.

In game we have 6 different classes: Healer, Tank, Hunter, Mage, Swordman, Rogue. Each class has different scale of abilities. Tank has good defence, but has small agility, Mage has small amount of HP, but huge attack etc.

Champions has level which represents his strength and for killing enemies he gain experience and levels. When champion gain new level his abilities rises.

Classes:

Ability – Class represents one ability like Health or Attack.

Abilities – Class represents collection of abilities. One of each type and can generate them on depend of Class.

Level- Class Level incudes Abilities, level and experience of champion. Can gain experience, rise levels and abilities.

Champion – Abstract class of champion cannot be instantiated. Has level actualHP image etc.

All classes (Tank, Hunter..) inherit from Champion and overrides some methods.

There are Tests for each of this classes, which tests most of their methods.

GameManager

It is the highest level of application, it controls what screen is currently showed. There is Class FireEmblam which is starting point of the game.

## Menu-

Character creation menu classes. Class names begins with ChC. Additionaly contains classes of old game main menu, stored only for potentialy later usage. There are no tests, because of highly graphical disposition.

### Menu-Elements

Graphical view of character creation menu.

### Menu-Model

Models for character creation menu.

### Menu-StartGame

This tag is only used when we debug the menu. It allows to show character creation menu elements individually.

## SideBar

In this tag, there are classes representing side menu for showing champion stats. And event log for showing the interaction. There are no tests, because of highly graphical disposition.

## UI-Elements

There is custom Enum for field type on the map. There is My event listener that handles events on the map.

## UI-Model

There is field struct that stores type of the field and position on the map and a Champion if present.

Field model stores Field struct and it represents one field on the map.

GameMap is the game model of the map, it handles all logic operations. Thanks to this you can set the map height and width also moving on the map. It interacts with SideBar and Event log. There are also tests for the map.

# Usage Example

Because our project is a game, we can’t do any example usages.

# Future update possibilities

* More maps
* Keyboard Control
* Saving and loading games to files
* Internet multiplayer
* Player vs. AI game mode
* Animations
* Even more fancy graphics
* Voice Acting, dialogs

# Reference to code with requested structure

* Polymorphism – in tag FireEmblamRpg classes like Rogue,Tank,etc as sublasses of Champion
* Singleton – at many places for example: in class GameManager/GameManagerModel method announcer and in class UI-Model/FieldModel method announcer.
* Observer – at many places for example: inclass GameManager/GameManagerElement method initialize
* Composite – We are using composite in many places in Bloc elements. For ex. we are adding BlTextElements, BlElements to another elements