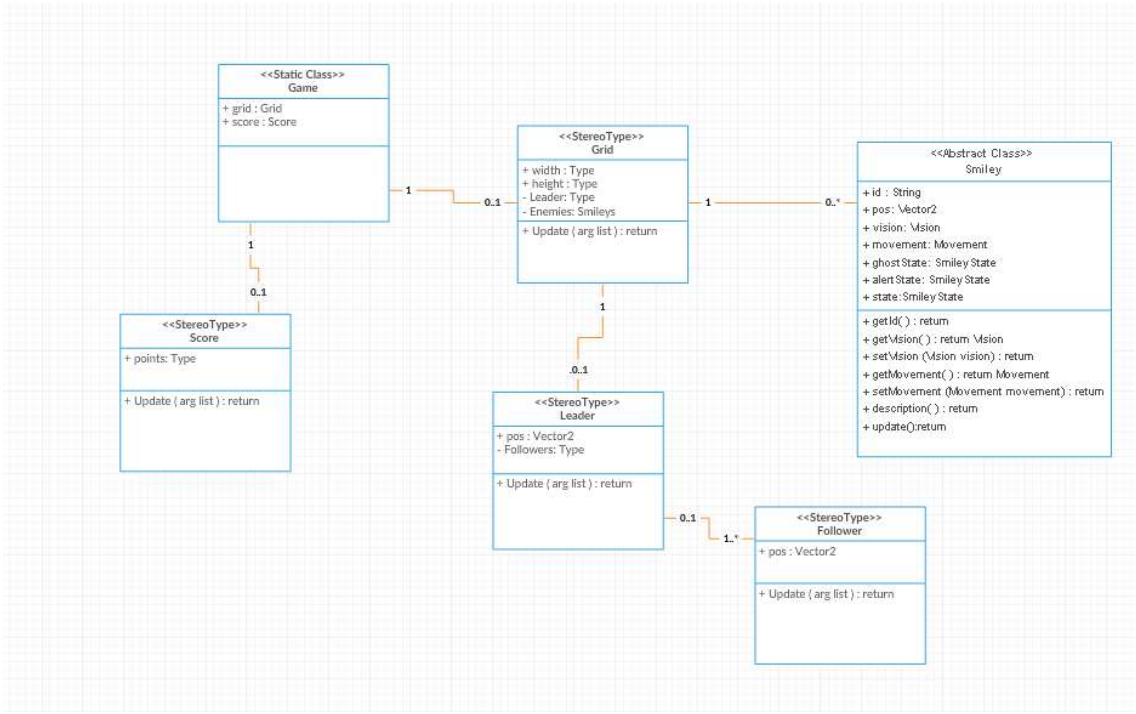
Class diagrams:

Singleton design Pattern for Game (overall structure)

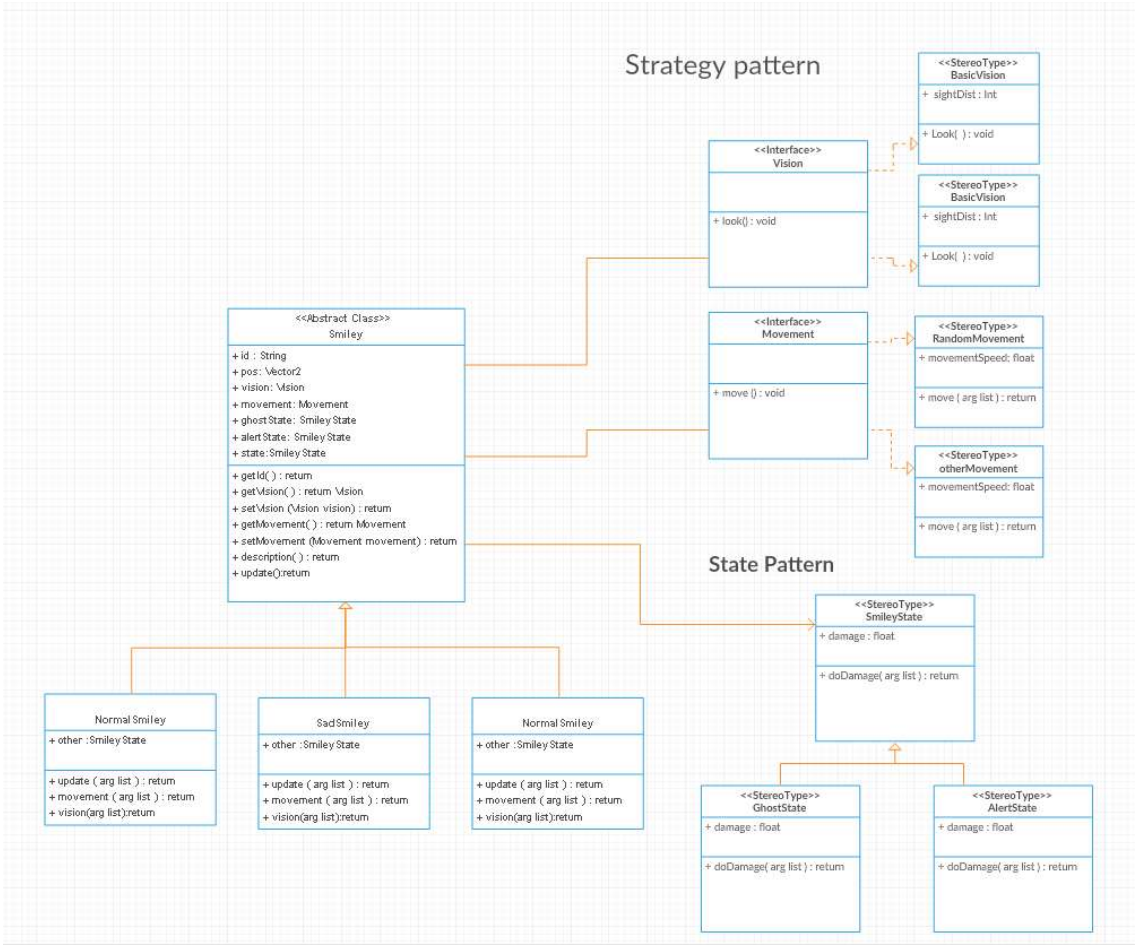


The creational design pattern the Singleton pattern is used to create Game object. Score stores current score allows for modification of score parameters easily on a game by game basis. Grid is the wrap around grid containing everything from the leader, his followers and enemies.

Singleton is used for…

- Ensure a class has only one instance, and provide a global point of access to it.
- Encapsulated "just-in-time initialization" or "initialization on first use".

Enemy class diagram in detail:



The enemies all extend from the smiley abstract class. To facilitate easy code reusability both the state pattern and the strategy pattern are used. The state pattern allows for code reuse of ghost states when the enemy is spawning or recovering and other similar potential changes in enemies. The strategy pattern is used to allow the enemies reuse movement and vision functionality. This allows the enemies to use each other's types of movement and switch from one to the other easily.

Behavioural Design patterns:

The Strategy Pattern:

- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it.
- Capture the abstraction in an interface, bury implementation details in derived classes

The State Pattern:

- Allow an object to alter its behaviour when its internal state changes. The object will appear to change its class.

- An object-oriented state machine
- wrapper + polymorphic wrapper + collaboration

Agile methodologies and planning

Week 1:

Research and idea formation

Week2:

“ ”

Week3:

“ ”

Week4:

Presentation on preproduction process showing commitment to the idea

Week5:

Sprint: Implement creational design pattern singleton, see main class diagram develop wraparound grid

Achieved: singleton and grid required to fix leader movement

Week6:

Sprint: fix leader movement get user feed back

Achieved: leader and follower movement

User feedback: slow reaction to fling on android

Week 7:

Sprint: enemy class implemented strategy and state pattern

Achieved: both design patterns complete

User feedback: it still not pretty

Week 8:

Sprint: collision detection

Achieved: disappeared enemies require debug

Week 9:

Deliver presentation on work so far

Week 10:

Add texture atlas do unit testing

Week11:

High score file system and user test

Week 12:

Present final project and results from user feedback

Resources for future work:

https://sourcemaking.com/design_patterns