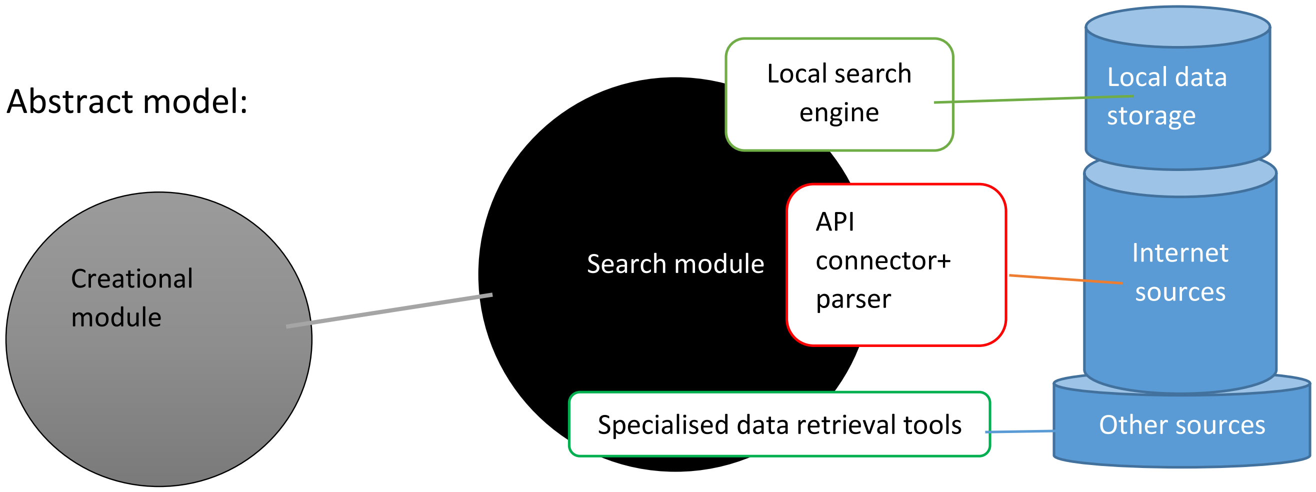
Goal1:

-make application capable of creating file with generated game logic/assets/

-make results relevant to information that can be found in the internet and variety of other information sources. Preferably result should relate on continuously-updating databases that represent general world of view of the society to obtain the highest value of player's understanding of game mechanics and gameworld rules.

Abstract model:



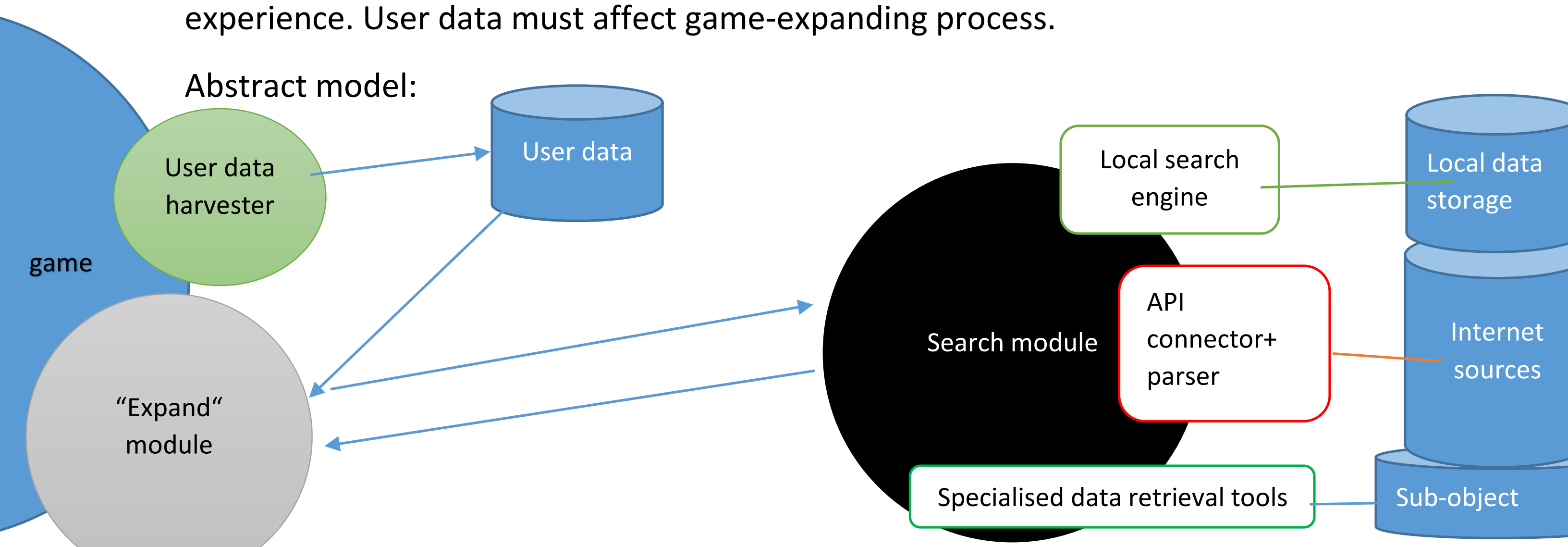
Usage :

1) Inherit search module class and modify it as you wish;
2) Create creational module(or derived) class and pass custom search module to it
3) Call createGame() method
4) …
5) Game file created.

Goal2:

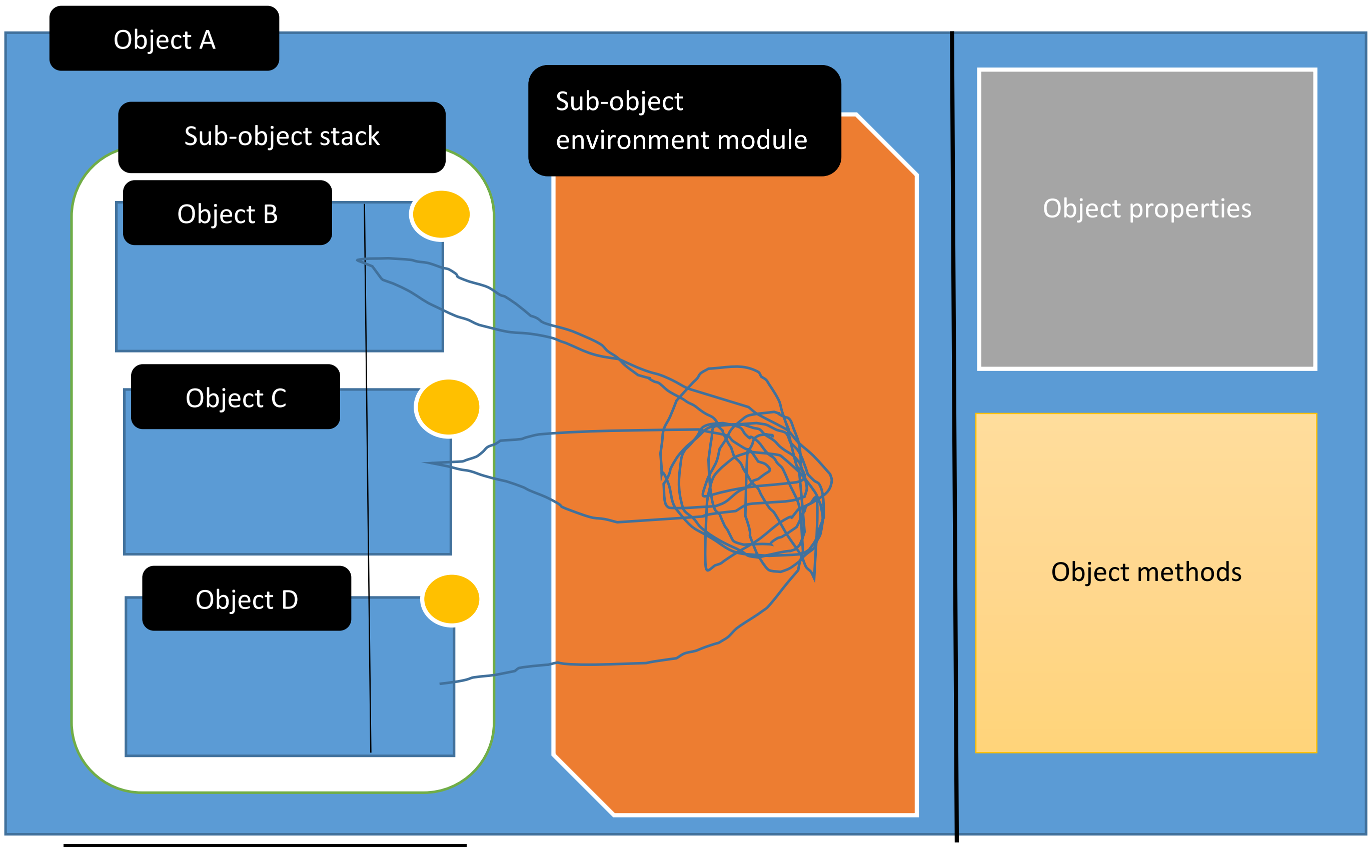
-make game capable of expanding itself by analyzing user preferences and game experience. User data must affect game-expanding process.

Abstract model:



Object model:

Every object in the game might sustain specific structure:



Sub-object environment module – component that describes sub-object interactions between each other and affects sub-objects in general. Implements sub-object environment.

Functions of sub-object environment:

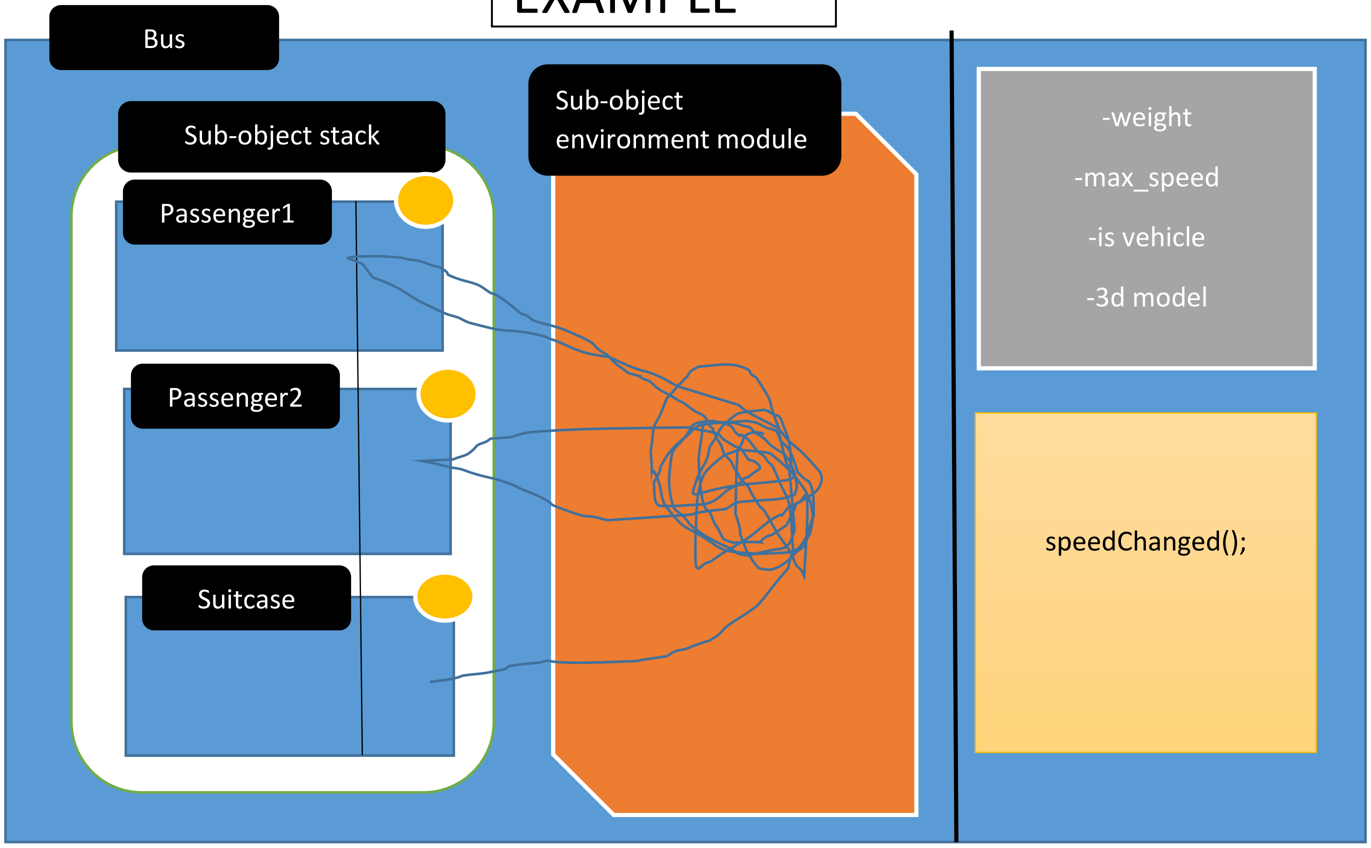
- Add sub-object (here constraints can be added to specify suitable sub-objects.
- Create sub-object
- Destroy sub-object
- Affect sub-objects (this functionality is completely custom)
- React to sub-object action

Methods and properties modules are not constrained (yet constructor/destructor might be needed);

To define object you need to define environment module, properties and methods.

Yellow circles symbolise object wrappers that add essential fields for sub-objects for them to be integrated into Sub-object environment.

EXAMPLE



Here **Passengers** and a **suitcase** were created inside a Bus object via **<u>sub object environment module(SOE-inner)</u>** or imported from some other object through **<u>sub object environmental module</u> method.**

In *addObject* and *createObject* we can perform needed checks to be sure that neither of inner objects has volume exceeding 2 m^3 .

When we created/added inner objects we can start our creation. Remember that bus itself is sub-object of some larger object (level, world, location) and is connected to larger sub-object environment there. Let's call that environment SOE-outer.

So, the game begins. Passengers are sitting on their sits (their position is determined by SOE-inner). Suitcase has no sitting state so it is positioned in between the rows.

Suddenly SOE-outer harshly stops the bus in outer environment. It's position is stopped and function speedChanged(delta_v) is invoked (signals-slots pattern).

SpeedChanged(delta_v) determines that when delta_v is over some @maximum_affordable_value it invokes other function that uses sub-object environment functionality to move inner objects and make non-sitting objects fall. Suitcase has no property as is_fallen but it is needed to visualise bus situation, so this property is stored in object wrapper of the suitcase. Finally, after suitcase fell changeSpeed() functions invoked for all three bus sub-objects to ensure that they would react on this event properly.

Tools to use:

For the Search module:

- Wikipedia api (unlimited)
- Google custom search api (100 queries/day subscription)
- Google web search api(deprecated, may not work)>>google reCaptcha can be hacked via google speech recognition api!{that can be used up to 60minutes/month}
- Google just parse html(can't be done through js in browser, can trigger captcha
- Yahoo BOSS(dead)
- Yahoo YPA (complicated & slow)
- Bing search api (now inside azure, limited use for 30 days)
- Azure cognitive(computer vision, speech to text, etc -> https://azure.microsoft.com/en-us/try/cognitive-services/ , needs subscription, has bounds, some services are given for first 30 days)
- Faroo api (free, slow indexing process {iphone->iphone4s, dwarf dwellings ->none} ,still working)
- http://commoncrawl.org (looks like the magic potion made exactly for data crawling, free)

----- language specific tools(parsers harvesters crawlers)

Python:
= https://elitedatascience.com/python-web-scraping-libraries
Java
= JSoup/Jaunt (parser + harvester)
C++
= curl
C#

Conclusions:

Goal: make framework with at least 2 modules(search and generation)

Search module might be flexible to change search api or better use several of them in the same time.

Search module might be using existing parser if there is one efficient enough to get needed data(mostly get rid of html tags and obtain data from specific components)

Throughout the development such things will be used a lot:

1)web crawling/scraping

2)parsing html/js/other data.

3) efficient containers, hybrid data structure design.

4) neural networks, parallel computing etc.