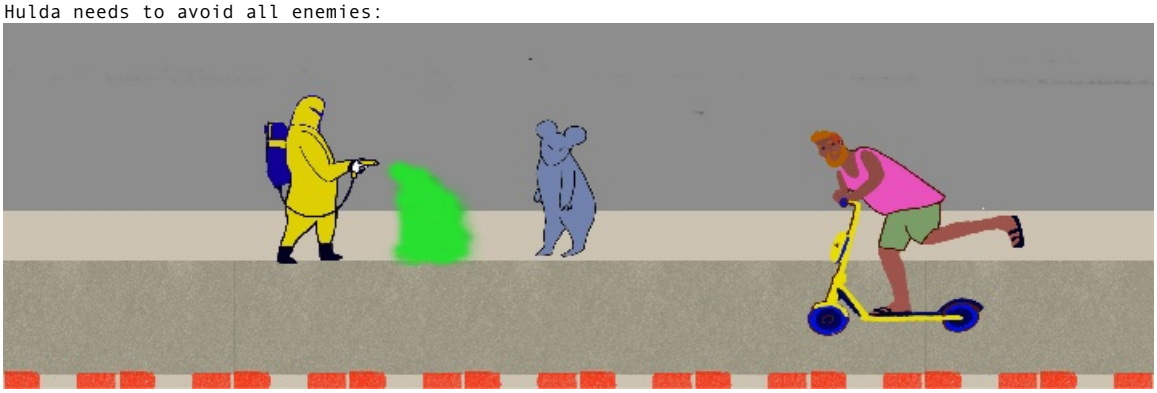
Ratata

Object-Oriented platform game by:
Bar Ifrah
Sharon Levi
Oren Holtzman

About

This is a story of 'Hulda', a Tel-Avivian sewer rat.

Being kicked out of her home due to Metro train construction in Tel Aviv, Hulda went out on the streets, looking for food to survive. But, there are many dangers in the city, such as: getting run over by a scooter, eating poison and having the city exterminator on your tail.
To survive- Hulda needs:
Hulda needs to avoid all enemies:

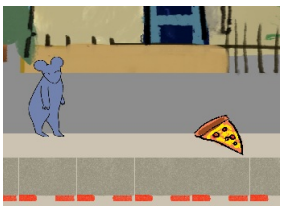
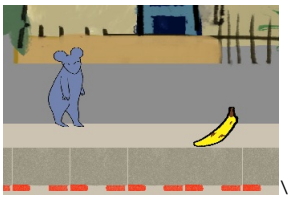


and avoid the poisonous apples:

\

to gain powers, Hulda needs to eat some burgers!:

to gain some points, Hulda needs to eat pizza, watermelon and bananas:

\

To move to the next level, Hulda needs to go inside the 'teleport' trash can:

Hulda starts with 3 lives. Every 100 points collected- hulda gets an extra life.
Hulda can gain lives from the 'special force' burgers. no more than 6 lives can be collected.

Create levels:

creating new levels by editing the 'Levels.txt' file.
level format is 20 rows height, 100 columns width. (can be seen inside 'Levels.txt' file.
symbols are defined in "macros.h" file.

Files and descriptions

CPP's + Headers:
To build a game object:
PhysicsObject.h/cpp
GameObject.h/cpp

Static Game Objects inherit from 'StaticObject.cpp':
Adanit.h/cpp
Food.h/cpp
ToxicFood.h/cpp PortalTrash.h/cpp Trash.h/cpp
RegularFood.h/cpp
SpecialFood.h/cpp
Road.h/cpp

Dynamic Game Objects inherit from 'MovingObject.cpp'
Enemy.h/cpp
-Exterminator.h/cpp
-Scooter.h/cpp
Player.h/cpp
DynamicFloor.h/cpp

Game controlling classes:
Controller.h/cpp
Stats.h/cpp
Board.h/cpp
DataReader.h/cpp
Resources.h/cpp
HighScores.h/cpp

Collision management: Collisions.h/cpp Listener.h/cpp Utilities.h/cpp Menu.h/cpp main.cpp

Implementation:

Algorithms we used:
A 'chaser'- the exterminator- chased Hulda in the X axis.
std::sort from STL.

Data structures we used:
2D vector for storing the current level objects.
Unordered maps- in 'Resources' class- to control the resources and extract them in O(1) time complexity.
Used unordered map to keep track of objects in the board, and assign them with unique ID's.
Used vector of (string, int) pairs- for names and scores in 'HighScores' class.

Design Patterns: Resources & Music - Singelton classes.
Used MultiMethods for manage collisions between objects.

Compile and run:

With IDE's:

Import all source code to your favorite IDE (works cross platform).
Compile and run.\

If you choose to run via CMD or Terminal:

Unix machines:

Go to either *Cmake-Build-Debug/Hulda* or to *out/build/x64-Debug (default)/Hulda.exe*
Open Terminal window.
Run:

```
% ./Hulda
```

Enjoy!

Windows Machines:

Go to either *Cmake-Build-Debug/Hulda* or to *out/build/x64-Debug (default)/Hulda.exe*
Double click Hulda.exe and play!

GitHub Repository

https://github.com/BarIfrah/Hulda-FinalProject

LinkedIn

Sharon Levi
Bar Ifrah
Oren Holtzman

Credits

Game design:

Shaked 'Keti' Zahor (instagram)

Physics:
Box2D Physical engine:

Graphic implementation: SFML

Game ScreenShots: