DICE Invaders – Coding Challenge

Download all the files you need here:
https://drive.google.com/drive/folders/1IJ3ECV6ClmnhoaluG9Eak3-Oo45qjvg4?usp=sharing

Create the game "DICE Invaders" using the supplied library. This assignment is the basis for the discussions at the technical interview.

Please read and consider the following instructions. Treat the assignment as you would a task given to you in a working environment.

Rules of the Game

Description
DICE Invaders consists of four different objects: the player ship, the aliens, the aliens' bombs and the player's rockets. The goal of the game is to shoot as many aliens as possible without being hit by them or their bombs.

The player ship
The player ship is positioned at the bottom of the screen. It can only move horizontally. It can fire rockets at the aliens. The ship can take three hits from aliens or their bombs before it is destroyed and the game is over.

The rockets
The rockets travel straight up. If they hit an alien, the alien and the rocket are destroyed, and the player's score is increased. Rockets cannot destroy bombs. When the rocket reaches the top of the screen, it disappears.

The aliens
The aliens start out at the top left corner of the screen, organized in rows and columns. They move slowly to the right, until the rightmost alien reaches the border of the screen. Then, they all take one step down, and start travel to the left instead. If the alien collides with the player, the alien is destroyed and the player's health is decreased. The aliens drop bombs randomly.

The bombs
The bombs travel straight down. If they hit the player, the player's health is decreased and the bomb is destroyed. A bomb disappears when it is outside the bottom of the screen. The bombs don't hurt other aliens.

The end of the game
If the player is hit three times by aliens or their bombs, or an alien reaches the bottom of the screen, the player loses and the game is over. If the player kills all aliens, a new alien army should be created at the top of the screen.

Compiling the Game (Windows)

Basics

- Launch a Visual Studio command line environment (with nmake.exe and cl.exe in the path)
- CD to the unzipped assignment
- Run `nmake` to build the code
- Run `DiceInvaders.exe`

Additional Build Targets

- `nmake` - builds a debug version of the game
- `nmake release` - builds an optimized version of the game
- `nmake dev` - build a debug version and opens up the game in the visual studio environment.
- `nmake clean` - cleans up the directory from intermediate files

Compiling the Game (Linux and macOS)

(Only tested on Ubuntu 16.04 LTS and macOS 10.12)

Basics

- CD to the unzipped assignment
- CD into the "sdl" directory
- Make sure you have SDL2 installed
- Linux: `sudo apt-get install libsdl2-dev` (depending on distro)
- macOS: Install the development libraries from [libsdl.org](https://www.libsdl.org/download-2.0.php) into `/Library/Frameworks`
- Run `make -f makefile.sdl` to build the code
- Run `./DiceInvaders`

Additional Build Targets

- `make -f makefile.sdl` - builds the game
- `make -f makefile.sdl clean` - cleans up the directory from intermediate files

Implementation

`DiceInvaders.cpp`

- Contains scaffolding code to get you started. Add as many files as you see
- fit, but make sure the project can be compiled using nmake as described above.

`Engine.h` (Do not edit!)

- The library interface. Should be self-documented.

`Engine.cpp` (Do not edit!)

- The implementation of the library.

`makefile`
- Simple makefile for compiling the game. Keep this makefile up to date if you add files to the project.

`makefile.sdl`
- Makefile for compiling the game on macOS and Linux with SDL support.

Target Hardware

Assume you are targeting hardware where:
- The CPU clock-frequency is low.
- Cache misses are very expensive.
- Branches are expensive. (No branch prediction).
- Memory is a limited resource. (No virtual memory).
- Memory allocations are expensive.

Quality Criteria

We will look at your software and consider:
- Clarity.
  - Is it easy to follow the flow of the code?
  - Is it easy to understand what the code does?
  - Do your comments add information?
- Simplicity.
  - Are there any unnecessary abstractions?
  - Are the algorithms used as simple as possible?
- Performance.
  - Will the code run efficiently on the target hardware?
  - For example, does the code make good use of the cache?

Prepare for the interview by knowing your code and be prepared to explain and elaborate on the decisions you make.

Remember that this is your chance to make an impression.

Feel free to deviate from the above, but remember to have clear reasoning behind your decisions. If you are applying for a specialist role, we also value if this shows or is reflected in your solution and argumentation.

What To Send To Us

- The game must build and work on targeted platform according to the instructions above.
- Engine.cpp and Engine.h should be without edits. The only exceptions are edits to the SDL platform layer to make it work on your operating system.
- Create and submit the project as a zip archive named `DiceInvaders_<your name>.zip`
- Don't include any executables (*.exe and *.dll) or other intermediate files (do a `nmake clean` or `make -f makefile.sdl clean` before zipping).

Bugs

Please mail bugs found in Engine.h/Engine.cpp to jonas.kjellstrom@dice.se.

Please note that Frostbite will never, in our own products, use any code, artwork or written work from practical tests submitted to us as part of a job application process. This is to ensure that all products are generated from within the company and wholly owned by Frostbite. Any future products released containing gameplay functionality, artwork or design ideas bearing resemblance to work submitted by you in this application process will be purely coincidental and completely unrelated to your submission.