Task Description

Introduction

A Universally-Accessible Game (UA-Game) aims to allow uses with different interaction abilities and capabilities to play. In order to achieve this, it is necessary to allow the game to adapt (specialize) itself during run-time to reflect the interaction abilities of the user.

Considering the great variety of possible interaction needs, the implementation of a UA-Game should be very flexible. The must not adopt specific modal representations or inputs – after all, the interaction needs of a user with visual impairments are very different from those of a motor impaired user.

This task provides a guided guide to implement a very simple prototype of a UA-Game using UGE game engine. The implementation explores different approaches to make the game logic input-output (IO) free. At a later moment, this allows the developers to specialize the game IO to suit the interaction needs of a specific user or from a group of users[1].

After the IO-free implementation, the guide suggests[2] how to implement different player profiles to suit the interaction needs of users with a single interaction disability. The chosen profiles to illustrate the usage of the engine are:

- Average user;
- Visually-impaired user (low vision);
- Visually-impaired user (blind);
- Motor-impaired user;
- Cognitive-impaired user.

The profile for average users assume the implementation of a conventional game, with graphical presentation and common input devices, such as mouse and keyboard. The later profiles will be build extending the initial implementation to, progressively and iteratively, include the abilities of the other profiles. As the base game is IO-free, it is possible to focus on UGE features to tailor the run-time IO of the game to the users with less effort after the initial implementation.

Design

To illustrate UGE approaches, this guide describes the implementation of a UA clone of the game Space Invaders. A possible design for an UA Space Invaders clone can be found in http://www.gamasutra.com/view/feature/1764/unified_design_of_universally_.php. The design described in that papers uses IO-free high-level tasks to create a clone of Space Invaders. The tasks are illustrated in Figure 1.

---

[1] It is important to note that, the more a game design is, the more complex it will be to create a fully IO-free implementation of the game. For complex games, this is almost impossible.
Thus, it is important to keep in mind that UGE engine considers, at this moment, the implementation of simple games – such as Pong, Snake and Space Invaders – as proof of concept.

[2] The goal of this guide is not to create a fully accessible, perfect UA-Game. Rather, the goal is to describe how the chosen approaches and the architecture contributes to a flexible, IO-free game implementation with run-time tailoring.

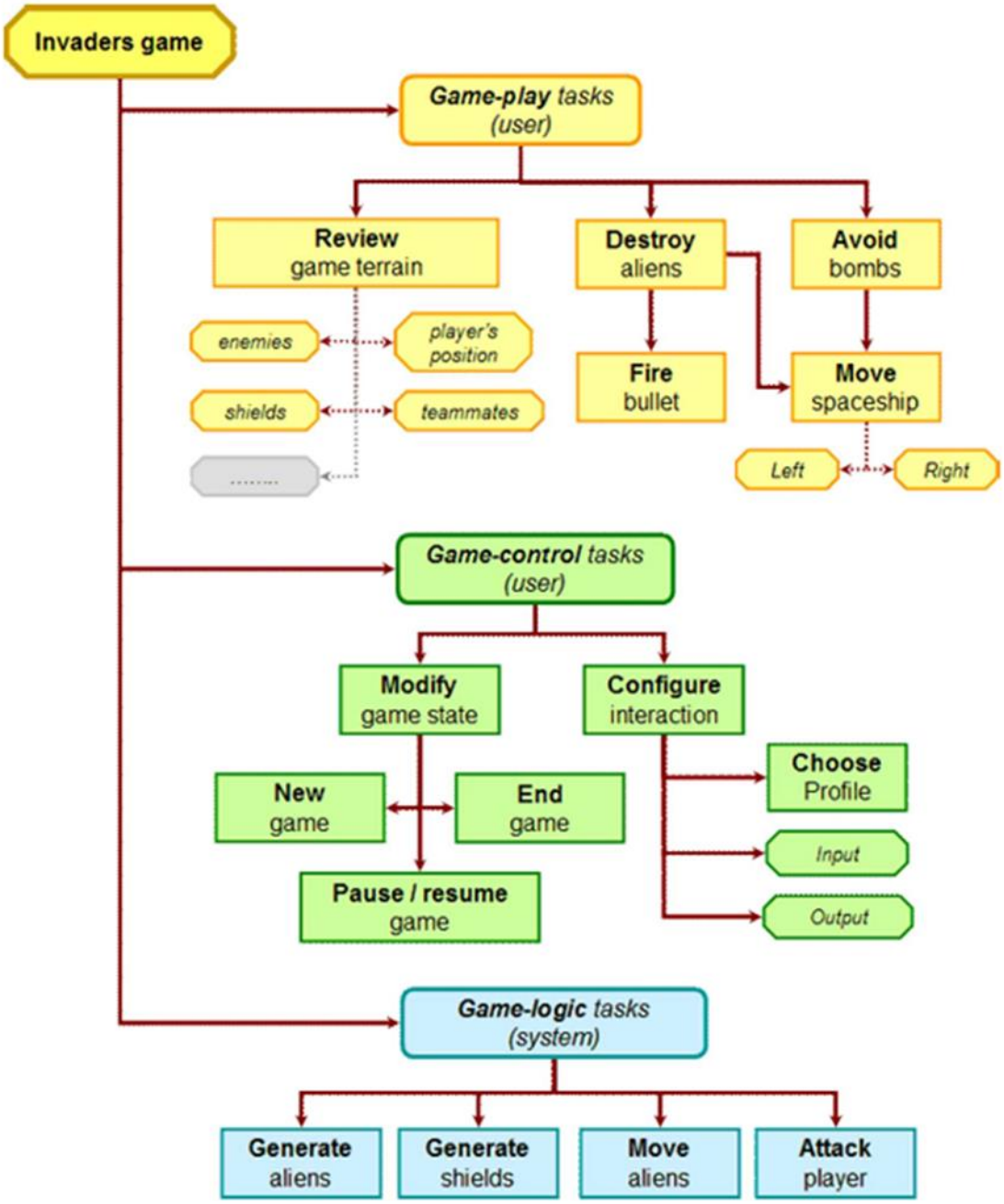

Figure 1. Tasks for an IO-free Space Invaders clone.

Anthony Savidis, Dimitris Grammenos.
Extracted from < http://www.gamasutra.com/view/feature/1764/unified_design_of_universally_.php>.

In Figure 1, it is important to note that the IO for the game are defined from a profile. This profile determines the IO specialization for interacting with game and to present the game world to the user.

Implementation

The implementation of the prototype, available and discussed at UGE's documentation (Section 7.2), describes a possible way to code the design illustrated in Figure 1 using UGE. To specialize the game, it considered the following profiles:

1. Average user:

    a. Traditional game implementation – it the initial implementation;
    b. Input: usual devices and commands;
    c. Output: graphics.
    d. Main UGE features explored:
        i. Components;
        ii. Events;
        iii. Game Commands;
        iv. Player Profile;
        v. Scene.
2. Visually-impaired (blind) user:
    a. Input: usual devices; optional automatic movement or firing;
    b. Output: sounds;
    c. Main UGE features explored:
        i. Components;
        ii. Events;
        iii. Player Profile.
3. Visually-impaired (low vision) user:
    a. Input: usual devices; optional automatic movement or firing;
    b. Output: graphics (using larger scale) and sounds (added after the blind profile);
    c. Main UGE features explored:
        i. Components;
        ii. Events;
        iii. Player Profile;
4. Motor-impaired user:
    a. Input: a single command (firing projectiles); automatic movement.
    b. Output: graphics;
    c. Main UGE features explored:
        i. Game Commands;
        ii. Player Profile.
5. Cognitively-impaired user:
    a. Different game speed;
    b. Different gameplay difficult;
    c. Input: usual devices and commands;
    d. Output: simpler stimuli.
    e. Main UGE features explored:
        i. Components;
        ii. Player Profile.

It is important to note that, at this moment, the game implementation is neither fully complete nor the game is totally accessible to any of these profiles. The implementation just describes strategies that, after more evaluations and iterations, would result into an accessible game. This decision aims to minimize influencing the evaluation result and it simplifies the approaches understanding, as it reduces the overall prototype complexity.

Task Description

This tasks consists on evaluating the architecture and the utility of the main approaches explored in UGE game engine:

1. Entities (actors) and components;

2. Events;
3. Game commands;
4. Player Profiles;
5. Scene.

Excepting the scene, all functionalities are detailed in Section 4 of UGE's documentation (the Developer's Reference). This documentation was provided together with this evaluation kit[3] and at:
<https://github.com/francogarcia/uge/blob/master/doc/Documentation.docx?raw=true>.

For a shorter introduction to UGE game engine and its approaches, it is recommended to read the UGE in a Nutshell guide, available with this kit and at:
<https://github.com/francogarcia/uge/raw/master/doc/UGE%20in%20a%20Nutshell.pptx>.

The flexibility provided by each of the numbered approaches is extended by using a data-driven architecture. This allows to customize the game using external configuration files (in this case, XML files).

For this task, we kindly ask for, in a game implementation and accessibility context, an evaluation of each of the numbered approaches. To do this:

1. Read the UGE in a Nutshell guide;
2. After reading the guide, open the Developer's Reference and refer to Section 7.2. From the start of Section 7.2 until Section 7.2.7, the documentation describes how to implement an IO-free Game Logic layer for the prototype.
   From Section 7.2.7 to the end of the document, the documentation describes how to specialize the game using UGE approaches.
3. It is possible to either use the final code from the repository or to follow the implementation with the extra material in this evaluation kit.
   For the first case, the code history from UGE's repository can be consulted to visualize the differences to the codebase with the insertion of new profiles. They are the updates from April 02, 2014 (<https://github.com/francogarcia/uge/commits/f2653e8fd2c1bde760b255cb5095fdac831267ab>).
   For the second case, it is possible to overwrite the current files gotten from the repository with the files from the extra material. Another option is to download the commit data using git. It is important to note it is necessary to change both the source file and the game data files. The extra material is available at:
   <https://github.com/francogarcia/uge-evaluation/releases/download/v2-Evaluation2/UGE-source.zip>.

At the end of the evaluation, we kindly ask to submit the answers to the online form at:
<https://docs.google.com/forms/d/18Mrf4MEEK2m7R4tpL9x9ffUQ-Gs9ONRtTdDYRJhN1gE/viewform>.

[3] The PDF version has some missing images for unknown causes. I apologize for the inconvenience. The DOCX version does not have any problems.