

'Hidden Object'

IMPORTANT

Due to a governing confidentiality agreement, we've refrained from disclosing actual client and solution names.

These have been changed appropriately, to more generic sounding terms and nomenclatures.

Industry:

Mobile Gaming

Platform:

iPhone

Business Challenge:

"Hidden Object" is a game in which a player is presented with a list of various objects and a graphical scene containing them. Player of this game is required to look carefully through the scene and select those objects in the list. Typically, there are series of scenes connected by a storyline.

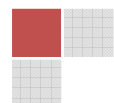
Software Solution:

As developers of this game, Rapidsoft has created an engine for this game which processes the input content and work as an Engine to allow multiple games to be easily built. Provisions of iPhone specific features like gestures, orientation, accelerometer and even the microphone and location awareness was mandatory. iPod feature have also been considered. This game was to be deployed in multiple countries so i18 support has also been built into the engine in the beginning.

The engine is concerned with parsing and applying content from templates at runtime, which will determine game flow and presentation. The content (including scene graphics, object location, storyline, music, sound effects, animation clips) was created separately and applied as and when necessary. This helps us to ensure a proper separation of engine logic and specification presentation.

The engine assumes this is the general game layout and flow for any hidden object game and therefore exposes a set of templates that can describe the details particular to an individual game. To create a new game, the template files are filled in, content is loaded into the iPhone project and the game engine is run. Depending on how the template files are filled out and what the content files contain, the runtime experience will be a different game.

The game engine developed by us did not generate code or executables - it is a runtime engine only. So if we want to have 2 different games, we have to copy the code base and start a new xcode project. The difference will be in the template files and content files that are contained in the xcode project. The code files are identical.



Features:

Game can be described by using 2 kinds of template files: “.hos” files (Hidden Object Scene file - HOS) and a “.hog” file (Hidden Object Game file - HOG). These are both simple text files, containing lines of name=value pairs. Many of the items in these lines will refer to other content files - like images, movies and sounds.

There is only one HOG file, which describes the scene order, storyline, and other items at the game level. There are many HOS files, one for each distinct scene in the game. The HOG file describes the overall sequence, game play, storyline and even branding of the game. The HOS files describe a scene’s layout and all of the objects within.

At runtime, the engine reads these files and constructs the game in memory and executes it according to the instructions in these template files.

Localization is straightforward using this format. The .hog and .hos files can be scraped and sent to translators – the resulting files can be bundled in locale-specific directories underneath the top-level directory. In this manner, the same binary could be used for multiple locales.

iPhone-specific features - specifically, the accelerometer and the microphone. These actions are described in the HOS file and are only possible when the scene is fully zoomed in and the actionable object is in the view. The full sets of interactions are as follows:

Tilting: the player flips forward (or backward) the phone at least 75 degrees on the X-axis in a span of less than 1/3 of a second. For example, this action may trigger a barrel to roll.

Cranking: the player rotates the phone 360 degrees in either direction on the Y-axis in less than 1.5 seconds. For example, this action may trigger a bucket in a well to be cranked up to the top.

Twisting: the player twists the phone 360 degrees in either direction on the Z-axis in less than 1.5 seconds. For example, this may open or close venetian blinds.

Rotating: the player rotates the phone 90 degrees or 180 degrees in either direction, with no time-span requirements. For example, this may cause a painting to tilt - or a flag to flip upside down.

Shaking: the player shakes the iphone back and forth at least 4 times in less than 2 seconds. For example, this may shake the leaves off of a tree or disturb a pile of rocks.

Blowing: the player makes a blowing sound into the microphone. For example, this could blow out a fire or blow the leaves off of a tree. Note that this only works on an iPhone, not an iPod Touch.

The engine only begins listening to the accelerometer and the microphone when the scene is fully zoomed in. When an actionable object has the correct action applied to it, a sound is played in the scene changes in some way to portray the change that occurred. These changes are persistent across visits to the scene.

