Valkyria® Text Game Engine

Documentation

# Core principles

The Valkyria Engine makes it easy for game makers without programming experience to develop full-fledged retro text games without writing a single line of code.

In this chapter, core principles of the engine are described in detail.

## Rooms

Valkyria games are all based around the concept of *rooms.* A room most commonly models a real-world room, but could also be considered a general container for the player. In other words – a room is a game world entity in which the player can be located at any point in time.

A game always consists of *at least one* room. This is where the player spawns – *the spawn room*. This room always have the unique id 0, this cannot be changed. Apart from this, the spawn room can be setup just like any other room (more on how to setup rooms in upcoming chapters)

## Doors

Doors, without surprise, connects two\* rooms. In the demo version of the engine, doors are always both ways. In other words, if the player enters though a door, he or she can always venture back through it and end up at the same place, just as in the real world. However, in upcoming versions this might change to enable some mind bending (and probably very confusing) gameplay.

Doors can be locked by *keys* (one or more). Keys are either already held by the player or picked up by the player in the game. If a door starts as locked, the corresponding key(s) must be placed within the game world, to prevent unsolvable puzzles. The game world setup is validated on initialization to prevent this from happening.

Keys can be placed in *containers* or in the rooms themselves (as rooms are just a special kind of container).

## Actions

A fundamental aspect of the Valkyria Engine is the use of *actions.* In short words, an action in the game is a triggered response with the effect that something happens. A good and trivial example is the *DisplayText* action – it simply displays some text to the player.

An action is either *Global* or *Localized*. Global actions can occur at any time, in any room, within the game world. Localized actions however are strongly tied to some *context,* as an example some specific room.

See a complete list of built in actions **here**

## Trigger

A trigger is a game entity that causes an action to be performed. The trigger is often based on some user input. A standardized trigger is the single keyword “help” that when typed displays some customizable help text to the user. The keyword “help” is simply a trigger for the DisplayText action mentioned above.

## Conditions

Most triggers have *conditions* attached to them. If the conditions are met, the action is triggered. If not, an alternate action (FallbackAction) is triggered instead. Fallbacks cannot (as of the demo version of the engine) have attached conditions.

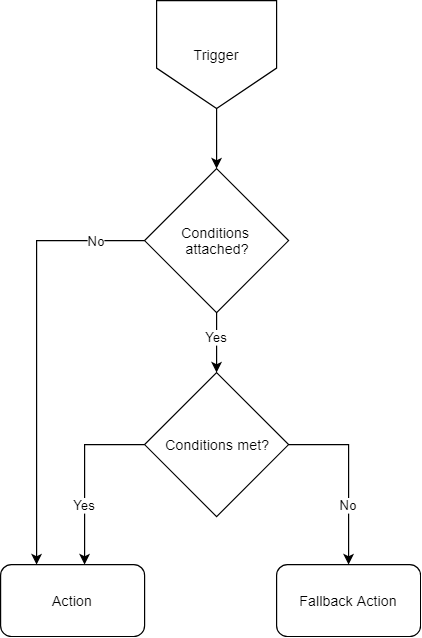
## Events

A game event is the combined process of a trigger, condition and action. An (conditional) event can be described simply by this language invariant function:

function Trigger()

If (conditions are met) then Action() else FallbackAction()

The following flow chart also depicts the whole event, with and without attached conditions



# A complete list of actions and their properties

## DisplayText

* Text (string)
* Path (xpath)

BeginDialog