Title: Wildlife Animation System

This wildlife animation system (WAS) is designed and implemented as part of TQ Engine. It allows game designers to put interactive wildlife into games.

Description

Current racing games that include animated animals are non-interactive. This means that animals in these games do not react to a player's action. For example, players in such games cannot pursue an animal.

The wildlife system creates interactive animals for racing games where players can observe animals' behaviour through high resolution animations, hear sounds they produced or chase them around. This is a new kind of game play experience that has not been explored before. Interactivity with the animals lets players feel that these animals are part of the game and not merely decorations. This enhances the realism of the game and makes the game more interesting. The wildlife system shows that with the use of graphics hardware and optimisation techniques, it is possible to implement a cost and resource efficient interactive and realistic wildlife system for 3D racing games.

The Features of WAS

The WAS has several great features. The figure below shows the overview of the WAS.

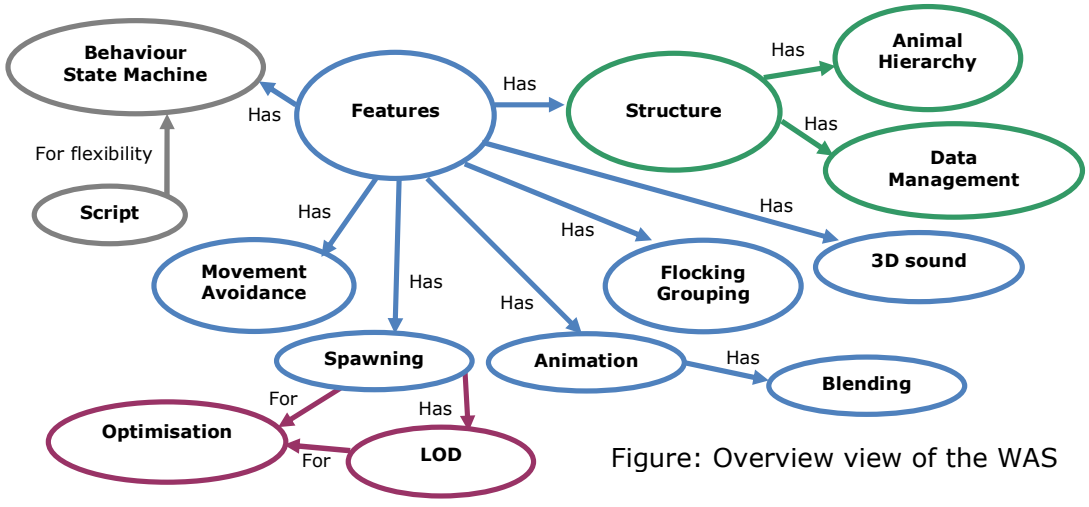


Figure: Overview view of the WAS

- Each animal has a behaviour state machine to control its behaviour. To simulate real behaviours, the animal transit from state to state. For example, walk state to graze state, walk state to run state, etc.

- To make the system more flexible, programmers or technical artist can script the states of the animals easily to the needs of different games. The text below shows a sample script and setting to configure an animal's behaviour state.

```
[Behavior1]

Name = "Idle"
AnimationType = "Idle"
AnimationFileName = "WLHare_Idle"
AnimationTimeScale = 1.000000
EnterScript = ""
ExecuteScript = "{ Idle; }"
ExitScript = ""
InitialBehavior = true

[Behavior2]

Name = "Walk"
AnimationType = "Walk"
AnimationFileName = "WLHare_Walking"
AnimationTimeScale = 1.000000
EnterScript = "{ DebugState; }"
ExecuteScript = "{ Walk; }"
ExitScript = ""

[Transition1]

Name = "IdleToWalk"
From = "Idle"
To = "Walk"
AnimationBlendDuration = 0.500000
Probability = 0.005000
TransitionLogic = "WildlifeTimeInState > 3.0 &&
Probability { BlendAnimation; PlayTransitionSound; }"
```

- The structure of the system allows animals to be highly configurable and manageable. The structure is further broken down into four parts namely animal hierarchy and data management.

  Animal hierarchy is the way all the animals are organised. WAS has three levels of animal hierarchy: One animal system consists of many animal groups and one animal group consist of many animals.

Similar animals share the same data from a single resource. This is to avoid data duplication.

- The movement avoidance system helps animal navigate around the world freely without knocking into objects. It also helps animal to flee from near threatening humans without getting hurt. This is one special feature about the WAS. It allows interactive of animal and player's vehicle but it prevents the animal from being ran over.

- Each animal has a 3D audio emitter. It helps player to locate the animal based on stereo sound emitted by the animal at different positions. This 3D audio also enhances game realism. For example, a player drives a car though a jungle will hear different kinds of wildlife sounds.

- Animals are group together based on types. This allows the animals to have group belonging and identity. The group of animals can move together, flock together and migrate together.

- Animation of the animal will be played according to which behaviour state it is in. Animation blending smoothes the animation transition preventing abrupt animation change from occurring.

- With the WAS spawning system, lots of animals can be simulated by using a few. This can be done by re-spawning the animals at the front of the player when the animals move out of the screen. When the animal re-spawns, it re-spawns from very small and slowly grows into their desirable size. This is to prevent the animal from popping out of nowhere. The spawning system helps to optimise the WAS, because the number of animals needed reduce greatly.

- Animal that is far away from the camera does not need to have high animation detail; therefore, it does not need to calculate its animations regularly. Using this information, animation optimisation can be realised. The animation of the animals is calculated based on their size and distance away from the camera. The further and smaller the animal is, the less frequent its

animations is calculated. Nevertheless, continuity and smooth animation can still be observed. This is known as animation Level-Of-Detail (LOD).

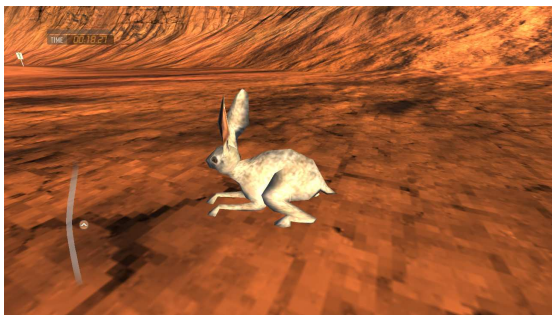
Hall of animals

Bobcat

A Cactus wren

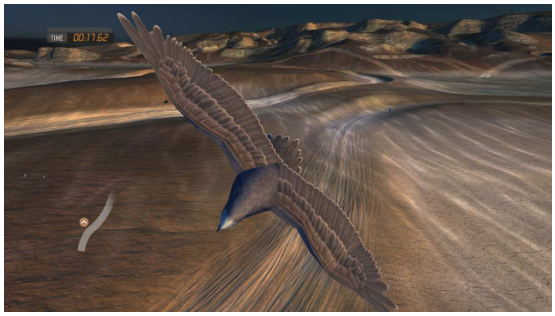
An Eagle

A Fox



A Hare

An Owl



A Hawk

A Javelina

A Roadrunner

A group of quails

A Squirrel

A Vulture