RTSUnitControl

version 1.2

Table of Contents

1. About

The RTSUnitControl is an asset contains some scripts and other tools for a RTS-like game development.

New:

- MiniMap

Features:

- Single selection and selection by a rectangle.
- Unit's attack and movement.
- Unit's AI – enemy detection and patrol area.
- Shader for easy creation RTS-like terrain.
- RTS-like camera with a smooth movement.

Ask any questions by e-mail: support@thoth-core.com

2. Get started

Load the sample scene from the *RTSUnitControl/DemoScene/Scene* folder to learn more about this asset.

3. Tips & Warnings

Tips:

- Set the material form the *RTSUnitControl/DemoScene/Terrain/Materials* folder as the custom terrain material to use our terrain shader.
- Use Layers to ignore the projection of the Projector material to some objects.
- Use Camera Culling Mask to ignore some layers by the MiniMap.

Warnings:

- Ensure that NavMesh was baked in your scene.

- Fill in the Material field in Projector components before using.
- Using of a large number of the active projectors may decrease the performance.
- The RTSUnitControl terrain shader uses the texture tiling from first texture for all textures in the Terrain → Texture list. It's limited by the 3.0 shader model.
- Ensure that the UnitControl.selectBox game object has the object with the Canvas component in its parents.

4. Scripting API

4.1. CameraControl

This class is used to smoothly move and rotate the camera.

4.1.1. Constants

float	**MAX_HEIGHT**	This constant defines the camera's maximal height above the terrain.
float	**MIN_HEIGHT**	This constant defines the camera's minimal height above the terrain.

4.1.2. Inspector fields

float	**height**	It's a current height of the camera above the terrain.
float	**movementSpeed**	It's the camera's movement speed by X and Z axes.
float	**rotationSpeed**	It's the camera's rotation speed by Y axis.
float	**scrollingSpeed**	It's the camera's movement speed by Y axis (It's used for the mouse wheel scrolling).
float	**movementDistance**	This member defines maximal movement distance of the camera from the start point.

4.1.3. Private fields

Vector3	**pivot**	It's the camera's start position. It's used to calculate the camera's maximal movement distance.
Vector3	**position**	This variable is used to calculate the camera's smooth movement.

Vector3	**rotation**	This variable is used to calculate the camera's smooth rotation.

4.1.4. Private methods

void	**CheckHeight**	This method is used to calculate the camera's height above the terrain.
void	**Move**	This method is used to calculate the camera's smooth movement. **Arguments:** *Vector3 direction* – the movement direction *float speed* – the movement speed
void	**Rotate**	This method is used to calculate the camera's smooth rotation. **Arguments:** *float speed* – the rotation speed
void	**SetHeight**	This method is used to smoothly change the camera's height above the terrain. **Arguments:** *float speed* – the height changing speed
void	**SmoothUpdate**	This method is used to apply changes of the camera's position and rotation.

4.2. Common

This static class contains several common methods and variables.

4.2.1. Input

bool	**touchScreen**	This property returns "true" if the device supports touchscreen and if the touch screen is used.
	Common	This constructor is used for the startup touchScreen property initialization.
Vector2	**GetInputPosition**	If touchscreen is used this method returns the touch position. Else this method returns the mouse cursor position.
void	**UseTouchScreen**	This method is used to enable/disable the touchscreen input.

Arguments:
bool use – is the touchscreen is used?

4.2.2. Math

bool	**Round**	This method returns “true” if the Vector2 “a” is round equal the Vector2 “b”. **Arguments:** *Vector2 a* – first vector *Vector2 b* – second vector *float accuracy* – the accuracy of the operation

4.2.3. Logic

Unit	**GetNearEnemy**	This method returns the nearest enemy unit or “null”. **Arguments:** *Unit unit* – the unit detects enemies *float distance* – the detection distance
Vector3	**GetRandomNavPoint**	This method returns a random point on the NavMesh. **Arguments:** *Vector3 position* – the position for the random point calculation *float distance* – the distance of the calculation

4.2.4. Screen

Rect	**GetScreenRect**	This method returns the rectangle in the screen coordinates from two points **Arguments:** *Vector2 start* – the start point *Vector2 end* – the end point

4.3. Laser

This is a unit’s weapon component.

4.3.1. Inspector fields

float	**rayLife**	It’s the time of the laser’s ray life.
float	**raySpeed**	It’s the laser’s texture movement speed.
float	**rayTilinig**	It’s the laser’s texture tiling by X axis.

4.3.2. Private fields

LineRenderer	**render**	It's a pointer to the LineRenderer component.
float	**fireExitTime**	This member is used to calculate the laser's ray life.
Transform	**target**	It's a pointer to the target of the ray.

4.3.3. Public methods

void	**Fire**	This method is used to attack by other components.

4.3.4. Private methods

Vector3	**GetHitPosition**	This is common method is used to calculate end position of the laser's ray.

4.4. LogicWarrior

This class implements a logic (AI) of warrior units.

4.4.1. Inspector fields

float	**attackDelay**	It's the time between two attacks.
float	**attackDistance**	It's the maximal distance for attack.
float	**damage**	It's a warrior's damage.
Laser	**laser**	It's a pointer to the Laser component.
bool	**detectEnemies**	Does the warrior detect enemies?
float	**detectionDistance**	It's the maximal distance for the enemy detection.
bool	**patrolArea**	Does the warrior patrol?
float	**patrolDelay**	It's the time between two patrol actions.
float	**patrolDistance**	It's the maximal distance to patrol.

4.4.2. Private fields

NavMeshAgent	**navagent**	It's a pointer to the NavMeshAgent component.
Unit	**unit**	It's a pointer to the Unit component.
float	**attackTime**	This member is used to calculate the time between two attacks.
float	**patrolTime**	This member is used to calculate the time between

		two patrol actions.
Vector3	**pivot**	It's the warrior's start position is used to calculate random patrol points.
Unit	**target**	It's a pointer to the target of the attack.

4.4.3. Events implementation

void	**Attack**	It's called when the UnitControl.OnAttack event occurs. **Arguments:** *Unit enemy*– the target of the attack
void	**Move**	It's called when the UnitControl.OnMove event occurs. **Arguments:** *Vector3 point* – the target of the movement

4.5. MiniMapCamera

This class implements the movement of the minimap's camera.

4.5.1. Inspector fields

Vector3	**followPosition**	It's a position relative to the follow target.
Transform	**followTarget**	It's the follow target.

4.6. Team

This is common class is used for the teams management in the game.

4.6.1. Constants

uint	**TEAMS_COUNT**	It's a maximal count of the teams.

4.6.2. Static fields

Team	**neutralTeam**	It's a pointer to the neutral team (0 team by default).
Team	**playerteam**	It's a pointer to the playerTeam (1 team by default).
Team[]	**teams**	It's an array contains all teams in the game.

4.6.3. Individual fields

Color	**color**	It's a highlighting color of the team's units.

int	**resources**	It isn't used in this asset version.

4.6.4. Contructor

Team	It's a static constructor for the fields' initialization.

4.7. Unit

This class implements basic features of units like as a health, a highlighting, UnitControl events implementation.

4.7.1. Inspector fields

unit	**teamNumber**	This field is used to define the team of the unit.
Sprite	**icon**	It's used to show the icon of the unit by UI.
Projector	**projector**	It's a pointer to the Projector component is used to unit highlighting.
float	**health**	It's the unit's health counter.
float	**maxHealth**	It defines maximal value of the unit health.
float	**regeneration**	It contains the value of the health regeneration.

4.7.2. Public fields

bool	**selected**	Is the unit is selected?
Team	**team**	This is a pointer to the team of the unit.

4.7.3. Events definition

OnUnitEventHandle	**OnUnitIsSelected**	It occurs on selection action. It's used by UI (UnitViewer component). **Arguments:** *Unit unit* – selected unit

4.7.4. Events implementation

void	**DeSelect**	It's called when the UnitControl.OnDeSelect event occurs.
void	**Select**	It's called when the UnitControl.OnSelect event occurs. **Arguments:** *Unit unit* – the selected unit
void	**SelectRect**	It's called when the UnitControl.OnSelectRect event occurs.

Arguments:
Rect rect – the selection rectangle

4.8. UnitControl

This class is used to control of units by input.

4.8.1. Inspector fields

Image	**selectBox**	It's used to draw select box for the box selection.

4.8.2. Events definition

OnEventHandle	**OnDeSelect**	It's used to deselect all units.
OnPointEventHandle	**OnMove**	It's used to set the movement target of the unit. **Arguments:** *Vector3 point* – the movement target
OnRectEventHandle	**OnSelectRect**	It's used for the box selections. **Arguments:** *Rect rect* – the selection rectangle
OnUnitEventHandle	**OnAttack**	It's used to attacks by player's units. **Arguments:** *Unit enemy* – the attack target
OnUnitEventHandle	**OnSelect**	It's used to select the unit. **Arguments:** *Unit unit* – the selected unit

4.8.3. Private fields

Vector2	**downPos**	It's used to calculate the select box coordinates.
Canvas	**selectCanvas**	It's used to calculate the inaccuracy of the select box.

4.9. UnitIcon

This is common class is used by the UnitView component.

4.9.1. Inspector fields

Unit	**unit**	This is a pointer to the unit. It's used to show the unit's icon and the unit's health.
float	**healthbarHeight**	It's a height of the UI healthbar in pixels.

4.9.2. Private fields

Image	**healthbar**	It's a pointer to the Image component of the healthbar.

4.10. UnitView

This component is used to show the selected units by UI.

4.10.1. Events implementation

void	**Add**	It's called when the Unit.OnUnitIsSelected event occurs. **Arguments:** *Unit unit* – the selected unit
void	**Clear**	It's called when the UnitControl.OnDeSelect event occurs.