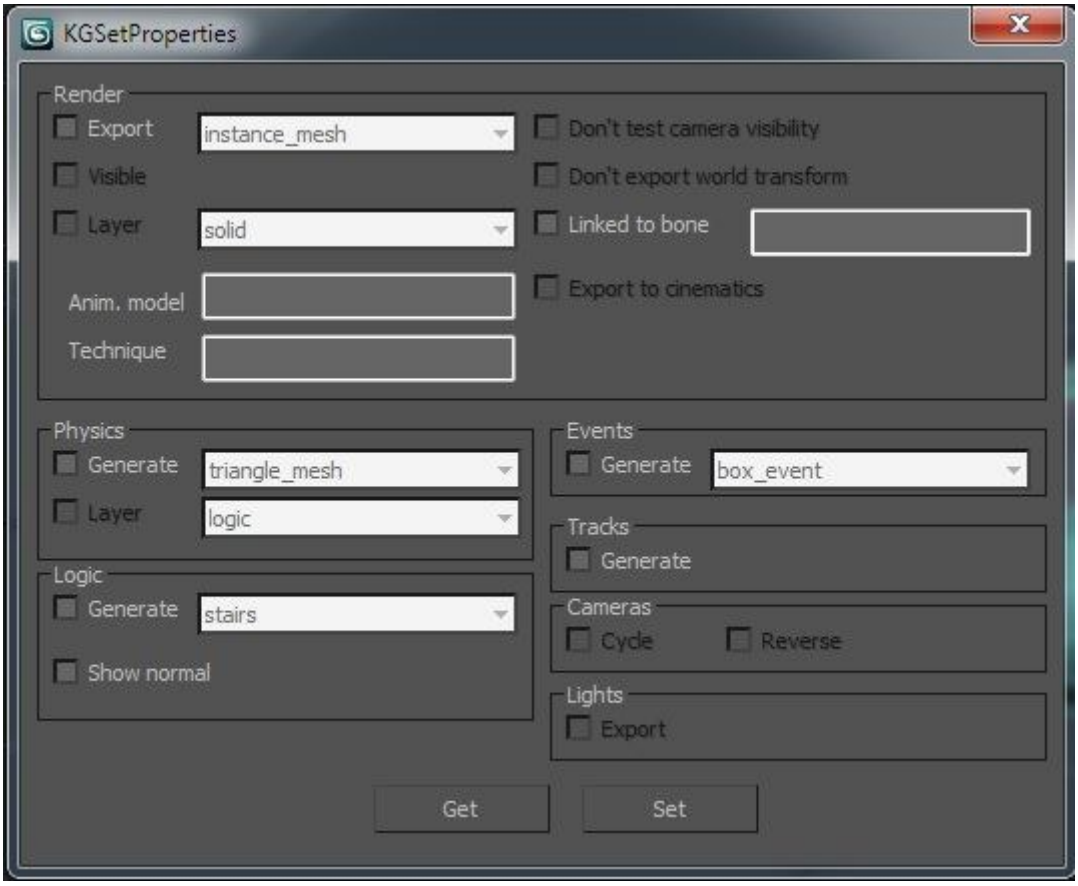
CINEMATICS

En este documento explicaremos lo necesario para poder exportar cámaras y elementos desde el 3D Studio MAX para poder generar pequeñas cinemáticas.

Exportar cámaras desde 3Dstudio MAX

Para comenzar utilizaremos nuestra utilidad de SetProperties realizada en MAXScript para poder exportar las cámaras, podremos asignarle los atributos de Cycle y de Reverse. El significado de los atributos son:

- *Cycle,* al terminar la animación la cámara vuelve a comenzar por el frame inicial realizando un ciclo de animación
- *Reverse,* al terminar la animación la cámara vuelve en sentido contrario a realizar el mismo recorrido



Una vez aplicamos las propiedades deberemos realizar la exportación de los parámetros de la cámara en cada frame clave generando un fichero xml como el siguiente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<camera_key_controller name="CameraMOV2">
	<key time="0.0" pos="-26.8859 9.80159 -6.46455" look_at="-20.3781 3.19479
8.47801" fov="0.602417" near_plane="1.0" far_plane="1000.0"/>
	<key time="4.66667" pos="0.963326 9.80159 -4.75075" look_at="-20.3781
1.67184 8.47801" fov="0.235158" near_plane="1.0" far_plane="1000.0"/>
	<key time="5.73333" pos="-12.2645 9.03728 2.90156" look_at="-20.3781 1.51945
8.47801" fov="0.199338" near_plane="1.0" far_plane="1000.0"/>
</camera_key_controller>
```

Para poder exportar los valores de la cámara o de un elemento deberemos ver en un frame determinado su estado, para ello utilizaremos un código en Maxscript similar al siguiente.

at time 12 current_pos = RHTranslationToLH obj.pos

Este fragmento establece la posición de un objeto en la variable current_pos en el frame 12, para poder generar el xml deberemos extraer todos los datos de la cámara durante los diferentes keyframes.

Para la gestión de la cinemática de cámara utilizaremos las clases que vemos a continuación.

Clase CCameraInfo

Además

```
class CCameraInfo
{
public:
	float				m_NearPlane, m_FarPlane;
	float				m_FOV;
	CPoint3D			m_Eye, m_LookAt;
	CPoint3D			m_Up;

	CCameraInfo();
	CCameraInfo(const Vect3f &Eye, const Vect3f &LookAt, const Vect3f &Up, float
NearPlane, float FarPlane, float FOV);
	CCameraInfo(CXmlTreeNode &atts);
};
```

Clase CCameraKey

Esta clase contiene la información de cámara de un frame determinado y el tiempo actual del frame.

```
class CCameraKey
{
public:
	CCameraInfo					m_CameraInfo;
	float						m_Time;
	CCameraKey(CCameraInfo &CameraInfo, float Time);
};
```

Clase CCameraKeyController

Esta clase contiene una cámara exportada desde el 3D Studio MAX en xml y nos permite gestionar el movimiento de las cámaras.

```
class CCameraKeyController
{
private:
	std::vector<CCameraKey *>		m_Keys;
	size_t					m_CurrentKey, m_NextKey;
```

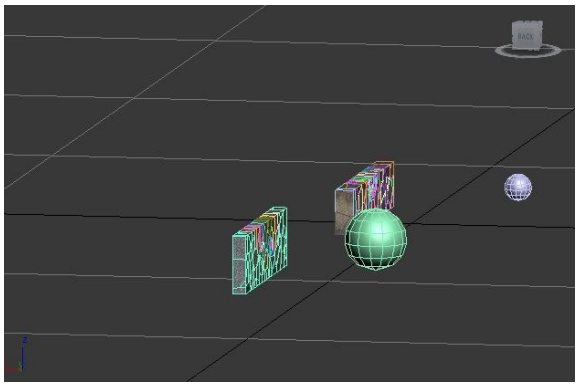
```
        float                                   m_CurrentTime, m_TotalTime;
        bool                                    m_Cycle;
        bool                                    m_Reverse;

        void LoadXML(const std::string &FileName);
        void GetCurrentKey();
public:
        CCameraKeyController(CXMLTreeNode &atts);
        void Update(float ElapsedTime);
        void SetCurrentTime(float CurrentTime);
        void ResetTime();
        float GetTotalTime();
        bool IsCycle() const;
        void SetCycle(bool Cycle);
        bool IsReverse() const;
        void SetReverse(bool Reverse);
};
```

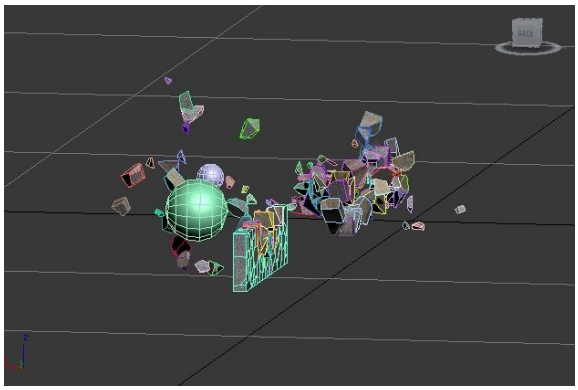
Esta clase define varios métodos, explicamos los siguientes métodos:

- *Update,* actualiza el estado de la cámara según el tiempo transcurrido
- *GetCurrentKey,* calcula el keyframe actual dependiendo del tiempo transcurrido
- *ResetTime,* establece a 0 el tiempo de la cinemática de cámara

Exportar elementos desde 3D Studio MAX

Además de poder exportar cámaras desde el 3D Studio MAX a nuestro motor, vamos a querer exportar objetos con movimiento, para ello vamos a crear diferentes clases que nos van a permitir generar cinemáticas como las que vemos en las siguientes imágenes.





Para poder realizar las cinemáticas de los objetos les estableceremos la propiedad de objetos cinemáticos, permitiéndonos la posibilidad de exportarlos dentro de una cinemática, generando un fichero xml similar al siguiente.

```
<cinematic name="DestroyWallComplete" duration="3">
	<cinematic_object resource="Box001_Part_6"
renderable_objects_manager="solid">
		<cinematic_object_key_frame time="0.0" pos="1.63108 0.564489 10.0025"
yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.0333333" pos="1.63108 0.564489
10.0025" yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.0666667" pos="1.63108 0.564489
10.0025" yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.1" pos="1.63108 0.564489 10.0025"
yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
	</cinematic_object>
	<cinematic_object resource="Box001_Part_7"
renderable_objects_manager="solid">
		<cinematic_object_key_frame time="0.0" pos="1.58712 0.945555 10.901"
yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.0333333" pos="1.58712 0.945555
10.901" yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.0666667" pos="1.58712 0.945555
10.901" yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.1" pos="1.58712 0.945555 10.901"
yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.133333" pos="1.58712 0.945555
10.901" yaw="-1.19209e-007" pitch="0.0" roll="0.0" scale="1.0 1.0 1.0"/>
		<cinematic_object_key_frame time="0.166667" pos="1.58414 0.949993
10.9059" yaw="0.00964391" pitch="0.000197887" roll="0.0115004" scale="1.0 1.0
1.0"/>
	</cinematic_object>
</cinematic>
```

En este fichero xml podemos apreciar que por cada objeto cinemático exportamos su traslación, rotación y escala para cada frame clave.

Para saber los frames claves de un objeto desde Maxscript podemos saber el siguiente frame clave y el anterior respecto a un frame. Para ello utilizaremos un código similar al siguiente.

```
key_frame=at time 1 trackbar.getNextKeyTime()
```

Este código nos devuelve el siguiente frame clave del objeto seleccionado después del frame 1, podemos utilizar el siguiente código para saber el frame anterior a un frame dado con un código como el siguiente.

```
key_frame=at time 1 trackbar.getPreviousKeyTime()
```

Cinemáticas en C++

Para poder implementar objetos cinemáticos dentro de nuestro motor vamos a implementar las siguientes clases.

Clase CCinematicPlayer

Esta clase nos va a permitir implementar un player de timeline.

```
class CCinematicPlayer
{
protected:
	bool m_Playing;
	float m_CurrentTime;
	float m_Duration;
	bool m_Cycle;
public:
	CCinematicPlayer();
	virtual ~CCinematicPlayer();
	void Init(float Duration);
	virtual void Update(float ElapsedTime);
	virtual void Stop();
	virtual void Play(bool Cycle);
	virtual void Pause();
	bool IsFinished() {return m_CurrentTime>=m_Duration;}
	float GetDuration() {return m_Duration;}
	float GetCurrentTime() {return m_CurrentTime;}
	virtual void OnRestartCycle();
};
```

La información que contiene y sus métodos es la siguiente:

- *m_Duration*, contiene la duración del player
- *m_Cycle,* nos dice si el player debe de realizar ciclo o pararse al final
- *m_Playing,* nos dice si el player está reproduciéndose
- *Update,* actualiza el tiempo actual y el reproductor de la cinemática
- *Stop,* para la reproducción de la cinemática
- *Play,* pone a reproducir la cinemática
- *Pause,* pausa la reproducción de la cinemática
- *IsFinished,* devuelve si la cinemática está o no terminada
- *OnRestartCycle,* es un método virtual que nos permite sobreescribirlo cuando creemos una clase que derive de esta para ser avisada de que la cinemática vuelve a comenzar el ciclo

Clase CCinematic

Esta clase deriva de RenderableObject y de CinematicPlayer, por tanto podremos añadir objetos de este tipo al manager de RenderableObjects y se acutalizará por cada frame.

```
class CCinematic : public CRenderableObject, public CCinematicPlayer
{
protected:
	std::vector<CCinematicObject *>		m_CinematicObjects;
public:
	CCinematic(MKeyValue &atts);
	virtual ~CCinematic();

	virtual void Stop();
	virtual void Play(bool Cycle);
	virtual void Pause();
	void LoadXML(const std::string &Filename);
```

```
        void AddCinematicObject(CCinematicObject *CinematicObject);
        void Update(float ElapsedTime);
        virtual void Render(CKGParams &KGParams);
};
```

La clase cinematic contiene los siguientes métodos.

- *LoadXML*, carga el fichero xml que hemos exportado previamente desde 3D Studio MAX.
- *AddCinematicObject*, añade un objeto cinemático al vector de objetos cinemáticos.
- *Update*, actualiza todos los objetos cinemáticos de nuestra cinemática.
- *Render*, este método estará vacío.

Clase CCinematicObject

Esta clase contendrá un objeto cinemático, derivará de la clase CinematicPlayer para poder controlar su tiempo y duración.

```
class CCinematicObject : public CCinematicPlayer
{
private:
        std::vector<CCinematicObjectKeyFrame *>  m_CinematicObjectKeyFrames;
        size_t                                   m_CurrentKeyFrame;
        CRenderableObject                        *m_RenderableObject;
public:
        CCinematicObject(CXMLTreeNode &atts);
        bool IsOk();
        virtual ~CCinematicObject();
        void AddCinematicObjectKeyFrame(CCinematicObjectKeyFrame
*CinematicObjectKeyFrame);
        void Update(float ElapsedTime);
        void Stop();
        void OnRestartCycle();
};
```

Esta clase contiene los siguienes métodos:

- *Constructor*, construirá el objeto cinemático, para ello cogerá un objeto renderizable según los parámetros del fichero xml.
- *IsOK*, devuelve si está correctamente creado el objeto cinemático.
- *Update*, actualizará el objeto renderizable estableciéndole la matriz calculada a partir de los key frames exportados desde el 3D Studio MAX.
- *OnRestartCycle*, este método sobreescribe el método virtual de la clase CCinematicPlayer y es llamado al reiniciar el ciclo de la cinemática.

Clase CCinematicObjectKeyFrame

Esta clase contiene la información de la traslación, rotación y escala de un objeto en un frame determinado, el cuál lo extraemos de la información del fichero xml.

```
class CCinematicObjectKeyFrame : public C3DObject
{
private:
	float 					m_KeyFrameTime;
public:
	CCinematicObjectKeyFrame(CXmlTreeNode &atts);
	KG_GET_SET_FLOAT(KeyFrameTime);
};
```