Record Keeping System Design Document, v2

Randy Angle

October 19, 1999

Introduction

There are two main uses for the record keeping system. The first and primary reason is to allow the game to track the progress of a child and tailor the activities and difficulty according to the lessons already finished. The second reason is to allow the development team to access records of the play during focus testing. These records will be used to improve the product as much as possible and identify weak areas of the interface or activity design.

The technique used to track progress is a combination of event, time, and activity completion recording. Minutes in certain zones or areas of the game will indicate which areas the child spends more time. Number of click events shows that the child is interacting and what they are interacting with. If a particular activity has some completion event then the game will record that progress.

The game design states that the Bricky and Clicky activities will select randomly from activities that the child has gained some experience with when choosing MATH, LANGUAGE, ART or MUSIC. Each activity will record the progress and increment a bucket indicating which areas the child learned more in.

Requirements

Because the Record Keeping system is used for both save status and focus test recording it must meet the requirements of both. The structure used to keep the save status will also have just enough additional information to make focus test records useful. It must record time, mouse events and activity events for all 31 activities (24 learning + 4 teaching + 3 interface). In any activity it will record click events on several types of scene objects (character, door, props before activity, and props after activity). In each scene the time before the formal activity and the time after the formal activity starts is recorded. These records will be saved to the hard drive on the user's machine and will be loaded when a child with a matching name plays again.

There are a few activities that require the ability to save an animation the child creates. Here we will record a string of delta time/event pairs that can be used to replay the animation created.

Structures/Classes

```
typedef enum
{
        RKEVENT_CHARACTER = 0,
        RKEVENT_DOOR,
        RKEVENT_MOUSECLICK,
        RKEVENT_MOUSECLICK_AFTER,
        RKEVENT_PROP,
        RKEVENT_PROP_AFTER,
        RKEVENT_MAX
} ERKEVENT;

typedef enum
{
        RKCURRICULUM_MATH = 0,
        RKCURRICULUM_LANGUAGE,
        RKCURRICULUM_ART,
        RKCURRICULUM_MUSIC,
        RKCURRICULUM_MAX
} ERKCURRICULUM;

typedef enum
{
        RKCHARACTER_KINESTHETIC = 0,
        RKCHARACTER_ART,
        RKCHARACTER_MUSIC,
        RKCHARACTER_SOCIAL,
        RKCHARACTER_LANGUAGE,
        RKCHARACTER_MATHEMATICS,
        RKCHARACTER_MAX
} ERKCHARACTER;

typedef enum
{
        RKINTERFACE_SCROLLING = 0,
        RKINTERFACE_SIGNIN,
        RKINTERFACE_TYPEIN,
        RKINTERFACE_MAX
} ERKINTERFACE;

typedef enum
{
        RKMODE_LEARNING = 0,
        RKMODE_TEACHING,
        RKMODE_INTERFACE,
        RKMODE_MAX
} ERKMODE;

typedef struct TAnimEventTag
{
        float   DTime;
        int     Object;
        long    Xpos;
        long    Ypos;
        TanimEventTag* Next;
} TAnimEvent;

typedef struct
{
        float   TotalSceneTime;
        long    TotalMouseClicks;
        int     Completions;
        int     Events[RKEVENT_MAX];
} TSceneRecord;
```

```
class CRecordKeeper
{
private:
        float   TotalGameTime;
        int     Learned[RKCURRICULUM_MAX];
        int     Character[RKCHARACTER_MAX];
        TSceneRecord   LearningScenes[RKCURRICULUM_MAX][RKCHARACTER_MAX];
        TSceneRecord   TeachingScenes[RKCURRICULUM_MAX];
        TSceneRecord   InterfaceScenes[RKINTERFACE_MAX];
        TAnimEvent*    pAnimSave;
        ERKCURRICULUM  CurrentCurriculum;
        ERKCHARACTER   CurrentCharacter;
        ERKMODE        CurrentMode;
        ERKINTERFACE   CurrentInterface;
        BOOL           CharacterClicked;

public:
        CRecordKeeper();
        ~CRecordKeeper();
        BOOL    Save(char* PlayerName);
        BOOL    Load(char* PlayerName);
        void    SetMode(ERKMODE NewMode);
        void    SetCurriculum(ERKCURRICULUM NewCurriculum);
        void    SetCharacter(ERKCHARACTER NewCharacter);
        void    AddSceneEvent(ERKEVENT NewEvent);
        void    AddAnimEvent(TAnimEvent* pNewAnimEvent);
        void    Update(float DeltaTime);
        void    HasLearned(ERKCURRICULUM LearnedCurriculum);
        void    Display(void);
}
```

Schedule Task List

System Tasks	Duration	Dependent
Design Record Keeping System	1 Day	Design Document
Code RK System	4 Days	RK Class designed
Integrate RK System	1 Day	RK Class coded
Test & Revise RK System	1 Day	RK System integrated
Rework #1 RK Class	2 Days	As Needed
Test & Revise RK Rework #1	1 Day	RK Class Reworked #1
Rework #2 RK Class	2 Day	As Needed
Test & Revise RK Rework #2	1 Day	RK Class Reworked #2
Total	13 Days	

Memory

The Record Keeping system uses arrays and lists to store the results of the save status. Because there are 31 scenes and the capability to store arbitrarily large animation event lists this seems like it could be quite a bit of memory. Each scene takes approximately 16 bytes for each (31*16 = 496 bytes). Each frame of an animation list takes 20 bytes; an animation 50 frames long would be 1000 bytes. A typical save game might reach 3k.

Risk Assessment

The real risk with the record keeping system is that we might need additional information to make useful decisions during focus testing. Every effort has been made to make it flexible and over-engineer the kinds of elements that we are recording.

In order to guard against faulty records magic numbers and checksums can be used in the files.

QA & Test

The Record Keeping system is a fairly straightforward file load, update and save process. It has all the faults that most file systems have when it comes to loading bad data. Testers should test and verify loads and try to feed the products bad data on occasion to see if it detects it. A function display the current save status to a debug console/window will help in the process of testing.