Lessons Learned from an ARTS / CS Game Design Collaboration

Torben Lorenzen Lorenzen@bridgew.edu
Department of Math and CS Bridgewater State College

Abstract

The first iteration of a Game Design collaboration between ARTS (modeling in Maya) and CS (scripting the Torque Game Engine) is described. The missteps of the collaboration are chronicled and a list is provided of the lessons learned to create a successful collaboration. A specification for developing textured game models with animations is included.

1. Game Design: Pre-Collaboration

The first semester that Game Design was taught (using the 500,000 line Torque Game Engine in a networked PC lab), the author proceeded cautiously and had the class download MilkShape avatars from the web and modify them. The games had no Bots and no movable models. Two students built a castle and a haunted house (modeled in QuArK) in which the class could stage team warfare with each student controlling his own avatar. Within this limited framework, each programming team of students succeeded in designing and implementing a game of their choice. Additionally each student was responsible for mastering a piece of software and creating an avi tutorial that was stored in a central library for present and future students to access. Thus each student became a consultant to the class in his area of expertise. The covered software included MilkShape (a modeler), CharacterFX (provides inverse kinetic animations), ms2dtsExporterPlus (converts MilkShape models to the dts format required by Torque), ShowTool (to debug dts models), LithUnwrap (skinning), Audacity and OpenGL (sound), QuArK aided by Python and Hammer aided by Wally (level design), and Terragen (sky box design).

2. Collaboration: The Plan

The second semester that Game Design was to be taught, the ARTS department was offering a digital modeling course (using Maya in a Mac lab). A collaboration was formed between the ARTS and CS departments in which the ARTS department was to model avatars and moveable models such as catapults and drawbridges and CS would write Torque script to control those models. The levels were to be created using Hammer and purchased components from www.Garagegames.com.

The initial four collaborators (an ARTS student, ARTS professor, CS student, and the author – a CS professor) met weekly for two months to brainstorm how to design the upcoming collaboration. The group decided to elaborate one of the earlier games set in the medieval/mythical past by expanding the rudimentary castles and including drawbridges, oaken doors, collapsing walls, cauldrons of boiling oil, and movable battering rams, catapults and siege towers. Since CS Game Design was a yearly offering and ARTS Digital Modeling was offered each semester, the collaboration would have to begin in two months or be postponed for an entire year. After the ARTS duo promised to create a model within a month, the author agreed to teach a collaborative version of his CS course with two months lead time.

3. Lesson One: Set the Collaboration Start Date

The artists produced a Maya Mac model a month later but when exported to the Torque Game Engine the animations were excluded. It took another month for the two artists to create a working export path (Mac Maya to PC Maya to maya2dtsExporter.mll) and the author's Game Design course began with one untested ARTS model.

Lesson 1: Do not schedule the start of the collaboration until ARTS has produced a prototyped model complete with unrefined animations and CS has executed those animations in the game engine. Completing these two milestones will demonstrate to each partner that the other is competent to begin the collaboration.

4. Lesson Two: Prescreen Models

The first assignment for CS was to learn the basics of the $35 MilkShape modeler and to prototype a textured model (with a root and a death animation) that would fit in a medieval world. It was acceptable for the prototypes to be quite crude and for animations to consist of only three key frames. Completing this free form weeklong assignment prepared the CS students to communicate with the ARTS students and produced two dragons (one beautiful; the other mistakenly described by the author as a plane), one Mario jumper, a snowman that threw snowballs formed from snowman belly, a drawbridge, a shield, a sword, a mace, a cauldron (sans boiling oil), a siege tower and a beautifully detailed battering ram complete with curved ram's horns. Interestingly enough, the best CS models were as good as later ARTS models.

Lesson 2: Prescreen the CS model choices to produce a higher useful yield of prototypes. Additionally, modeling simple weapons can be discouraged; they can be purchased as part of a medieval weapons pack at www.garagegames.com/pg/product/eula.php?id=76.

5. Lesson Three: Supply Model Spec Sheets

One of the CS students modeled his creation using a month-long-trial version of Studio Max and exported the model to MilkShape where he then added the animations. (ARTS did not want to follow a similar sequence with Maya). Screenshots of the MilkShape prototypes were passed to the ARTS students with the intention that the screenshots should serve as initial design documents for the ARTS students to use in preparing Maya prototypes. After all, a picture is worth a thousand words, right?

Lesson 3: Wrong! Supply a thousand words to accompany each screenshot. The professors must prepare a standardized spec sheet (see the sample at the article's end) for each of the approved models and these sheets must be distributed to both classes.

6. Lesson Four: Teach ARTS How to Prototype

Those promised Maya prototypes never arrived. Perhaps ARTS did not want to hand off "inferior" products to CS. Perhaps the artists felt that their models were 95% complete – that they were only one little tweak from perfection (Sound familiar?). Perhaps making prototypes and exporting them would have increased the workload for ARTS. In any case, CS sat on their collective hands for two weeks – not willing or not understanding how to write script to interact with nonexistent models that had shaky specs at best.

Lesson 4: The CS professor must schedule a two hour block of time to explain the necessity for prototyping to the ARTS students. The ARTS and CS professors must coax prototypes out of the ARTS students in this two hour span.

7. Lesson Five: CS Creates Complete Prototypes

Lesson 5: The first CS assignment must include three-key-frame versions of all the animations from the spec sheet and a model and animation export to Torque (using the ms2dtsExporterPlus from http://chumbalum.swissquake.ch/ms3d/download.html). In this manner CS can begin to test script code on their own primitive models and primitive animations while ARTS creates Maya model prototypes.

8. Lesson Six: ARTS Does Not Adequately Test

A month after the semester began, the beautiful ARTS models finally arrived. The exquisite goat-man avatar sunk up to his waist in the ground and some of his animations were off axis by 90 degrees and bullets flew through him. All avatars could walk through the new gorgeous catapult and its wheels were transparent. The projectile that the catapult tossed disappeared after firing. Avatars fell through the new drawbridge. The stairs of the siege tower floated ten feet above the ground out of reach of the avatars. The castle door had a great death animation, but no open or close animations. Because the door had no door frame, it was hard to seamlessly attach it to the castle. The battering ram's root animation was judged obscene by all who saw it. The sword's sheath animation was cyclical but no sword ever sheathed an enemy to death.

It was the professors' combined inexperience that was responsible for this situation. The author had downloaded working models from the web and had only done rudimentary modeling in MilkShape. The ARTS professor had modeled before but had no knowledge of the many additional requirements that gaming would require of the models. This blissful ignorance gave way to a semester long sequence of panicky fire fighting.

The models' failings were listed by the CS students as they incorporated the ARTS models into the Torque game but there was a possibility some failings could be a consequence of scripting errors. So case by case, it was necessary to work with the ARTS and CS students involved and to determine if the problem was a modeling problem (ARTS to fix) or a scripting problem (CS to fix). In either case the professor was often challenged by the offended student and had to prove to the student's satisfaction that it was his problem before he would honestly work on it. In those cases where it was a modeling problem, the ARTS students learned that their beautiful art did not work. Artists have heard the phrase "That doesn't work for me" as an aesthetic judgment but art not working in an absolute sense was a foreign concept to them.

Unfortunately testing, debugging and backing up were also foreign concepts to ARTS students. One ARTS student lost his Maya version of a beautiful drawbridge and didn't have time to recreate it. Several ARTS students did not do a "proof of concept" and spent weeks creating different avatar skeletons provided with one animation each with the untested (and incorrect) assumption that these single animations could later be shared amongst avatars with different skeletons.

Lesson 6: ARTS students are probably unaware of basic CS testing principles and will not magically master them in one month.

9. Lesson Seven: Create a Test Station for ARTS

ARTS did not adequately test their models so that work fell to the unwilling CS students. As the ARTS students fixed one bug in a model, a different one was sometimes introduced in the export process. So the increasingly resentful CS students were forced to participate in a debugging cycle with the ARTS students. And the ARTS students didn't appreciate this process one bit either. Things got so bad that some CS students asked to create the models themselves. In hindsight it made perfect sense that ARTS students would know as few CS testing concepts as students beginning a CS1 course.

Lesson 7: Create an ARTS test station in the CS lab. This station runs a "game" that is set up to accept all the models agreed upon on the spec sheets and to test their animations. The player avatar can shot at the models and tests collision meshes. By changing the distance between the avatar and model, Level Of Detail (LOD) can be verified. Together the two professors must test each model in this game environment and only certified models may be passed on to CS.

10. Lesson Eight: Create a Backup Plan

When the ARTS models were finally error free, each CS student wrote script to control one model. However there was not enough lead time to integrate these independent modules into a working game. This disappointed everyone and surprised ARTS. As modeling deadlines slipped, ARTS was repeatedly warned that CS required lead time to complete the scripting. ARTS had little knowledge of scripting and didn't understand why it took CS so long to take functioning models and put them into a game. The basic problem was that CS did not make a major scripting effort until ARTS completed their models. It is imperative that the modeling and scripting efforts proceed in parallel. Implementing the lessons included in this article will facilitate that required parallel development.

ARTS had counted on submitting a working game to the school's art show to showcase their models and to build interest in the new digital modeling course. This failing was a major blow to the strategic plans of the ARTS professor.

Lesson 8: There must be a backup plan in case all the models are not finished or a complete game is not produced. CS can reuse models from previous semesters or download internet models. ARTS can produce 3D prints of their models for a sculpture art show.

Model Specifications

Student Name: _____ Email: _____
Model and Folder Name: _____
Prototype delivery date: _____ Model delivery date: _____
Model and Folder Name: _____

In that folder, please place the following:
Maya Model Name: (ModelName.ma or ModelName.mb)
Torque Model Name: (ModelName.dts)
Animation Files: (??????????.dsq)
Texture Files (use .jpeg unless you use .png to achieve transparency)

Please check that the texture files are usefully small and each has a size of
8x8 16x16 32x32 64x64 128x128 or 256x256

Texture Name	Model Part that is textured	Texture Size

Animation Name	Description	Does it Cycle?
Root (required)		yes
Death (required)		no

☐ Polycount below 2000 triangles ☐ Animations tested in Torque Show Tool

☐ Collision Mesh (Col) tested in game ☐ Line of Sight Mesh (Col_LOS) tested

☐ LOD (Level of Detail) (detail2, detail64, detail 128) tested in game

☐ No concave surfaces (no dimples or hollows).
For example, a catapult arm must end in a closed drum – not an open spoon.