Texnh-II Initial Planning Meeting

Texnh-I Background

Perceived problem with existing curriculum -

Poor quality of the output... A majority of students in *senior* level courses were unable to complete the following: *Write a program in any language that will iteratively compute the sum of the first 1,000 integers (don't use n (n + 1) / 2) .*

Possible causes -

Unispiring assignments (grade averaging programs, phone book, inventory management, trivial games)

Focus on mastering complex API's and programming "environments" instead of design and implementation of fundamental algorithms (Drag and drop programming)

"Why should CS I/II students be able to implement searching and sorting algorithms when thats just what the xxxx class does?" "You don't need to have to understand how an engine works to drive a car.".... ad infinitem et nauseam.

The abstraction first paradigm

should integer addition be taught a special case of an abelian group? (abelian groups first or the "New Math")
or
should an abelian group theory be taught as an abstraction of integer arithmetic? (numbers (machine model) first)

The *texnh I* project

Proposal

The goal is a zero-based re-design of the *B.A. degree* program in computer science. The overriding directive of the new design will be the direct incorporation of DPA and computer graphics research results into all required computing courses in the curriculum. Instruction in these (new) courses will be strictly oriented toward *large-scale problem-solving, where the large-scale problems are those that have arisen naturally in the research investigations of the co-PIs*, each of whom has published in graphics or special effects.

Our (tentative) collection of key concepts comprises:

- a machine model (imperative programming, machine capabilities, machine limits)
- a connection model (networks for communication and distributed processing)
- software design (the object-oriented paradigm, large-scale development, testing)
- windowing and operating systems (resource management, protection levels, security)
- data structures and performance (performance measurement, bottleneck identification, work-flow management)
- cross-platform computing (PDAs, embedded systems, cross-compilation, external device control)

Motivated in part by the success of the DPA program in attracting minorities / women, but there is little evidence (so far) that this success extends to the CPSC undergraduate program.

Texnh I implementation

Considerable deviation from what was proposed.. much of this based upon the art of the possible.

Reality checks:

- Not enough students to justify a separate BA curriculum.
- Changing the entire CS/CIS curricula in one "swell foop" not feasible.

Hence, a nose-under-the-tent approach (though not actually perceived as such at the time by the camels)

- Pilot (*pre Texnh I*) course CPSC 215... C programming for students with 2 semesters of objects-first Java was converted from traditional problem domains to ray tracing *(no faculty approval necessary... just some camels who volunteered to teach it).*

Based upon demonstrated success of the pilot *(nice pictures and positive student feedback)* the NSF funded *Texnh I.*

Based upon demonstrated success of the pilot *(nice pictures and positive student feedback) AND* the NSF funded *Texnh I.*

- Faculty approved offering parallel tracks in CS I/II taught by Matzko / Davis in 05/06

Based upon demonstrated success of new CS I/II (more nice pictures, postitive student feedback, and some comparative analysis of student skills in both of the parallel tracks).

- Faculty approved new CS I, II, III, IV curriculum in spring 2006.

Elements of the new curriculum

Problem based learning in the visual domain (other domains? e.g. aural ... acutally proposed in Texnh I).

- Objective is to use *one* problem that is incrementally solved in the course of the semester.
- Not so easy to do in CS I.. but easier (to a degree) in other courses.

Current problem domains

- CS1 - some manner of image transformations (C)
- CS2 – raytracing (C / C++)
- CS3 - surface reconstruction -> photon mapping (C++)
- CS4 – a distributed board game (Java)

Projects typically have alternative implementations and score is based upon how much the student *got right* rather than how much s/he got wrong.

Managing faculty pushback

Common complaints

- This will be "Too much work" for faculty to learn new programming language / new problem domain.
- Not interested in "learning graphics".
- Not able to "learn graphics"... usually applied to others!!
- Focus on "graphics" distracts from basic algorithm development and software engineering techniques (To this day some steadfastly refuse accept that NO GRAPHICS APIs are in play... *possibly because their educational worldview is teaching how to use complex APIs??*)
- After one semester its over... Students will simply turn in last years project

Faculty mentoring

- A faculty mentor is made available to a faculty member who teaches a *texnh* course for the first time.
- The mentor and mentee teach different sections of the course (desirable but not mandatory)
- The mentor provides detailed lecture notes, assignments, and coordinates the lab
- The new faculty member may or may not actually attend the mentors lectures and may use as much or as little as s/he sees fit.
- The next semester the new faculty member teaches the course "on their own".

The *texnh II* plan is for Clemson faculty to assist UNC-W and WCU faculty in this way.

Personal experiences in Faculty mentoring in texnh courses

- Senior lecturer - CS 101
- Senior associate prof - CS 215
- Senior full prof - CS 102

In all three cases the faculty members with whom I worked expressed the following sentiments:

- they found the *texnh* approach a fresh an interesting way to teach the material
- because of the resources provided, there was no feeling of "too much work"
- all were enthusiastic about teaching the course again "on their own"
- the senior associate subsequently taught CS 215 several times and evolved his own spin on the raytracing material

(in CS 101 and 102 I was mentor the same semester I taught the course for the first time.... that IS a good deal of work)

The CS 2 (raytracing course)

A great diversity of approaches *(and thus deliverables)* are possible and obviate concerns on *the assignment recycling problem.*

For example:

- Sarah/Tim/Robert -- see dissertation as a starting point
  - Informal linear algebra
  - Model description language last
  - Teamwork encouraged

- Mike/Wayne/David -- see notes as a starting point.
  - Geometry based upon more formal linear algebra
  - Model description language first
  - Teamwork verboten

Variations on the raytracing problem

Coordinate systems (changing any one of these completely defeats the completely inept cheater)

- left handed / right handed
- positive y direction
- origin location

Visible entities (ray / surface intersection equation is quadratic for all qaudric surfaces)

- spheres
- ellipsoids
- cones
- cylinders
- parabaloids
- infinite planes
- bounded planes
- rectangular solids
- surfaces of revolution

Surface properties

- reflectivity (ambient, diffuse, specular)
- partial transparency (with or without refraction)
- color
    - solid color *(r, g, b reflectivities)*
    - texture mapped (with reflective properties)
    - tiled (fixed or arbitary orientations of the tiling)
    - arbitrary procedural shading (a generalization of tiling)

Light sources

- point sources
- conical spot lights
- rectangular spot lights
- cylindrical spot lights
- projecting textures

Refinements

- antialiasing
- soft shadows

Model description

A simple number and comment format

```
8 6             world x and y dims
4 3 3           viewpoint (x, y, z)

12               textured plane
0.0 0.0 0.0      amb, diffuse, spec
3.0 3.0 3.0      amb, diffuse, spec
0.0 0.0 0.0      amb, diffuse, spec
0 0 1            normal
-12 -10 -8.0     point

11               sphere
0.2 0.2  0.2     amb, diffuse, spec
11.4 2.6 3.6     amb, diffuse, spec
0.0 0.0 0.0      amb, diffuse, spec
6 3 -3.0         center
2                radius

10               light
18 18 18         emissivity
4 3 5            center
1                radius
```

An entity / attribute / value format amenable to generalized parsing

```
camera cam1
{
  pixeldim  800 600
  worlddim  8 6
  viewpoint 4 3 6
}

material white
{
   ambient 0 0 0
   diffuse 9 9 9
}

sphere earth
{
   material white
   center 4 3 -8
   radius 5
}

projector front
{
   emissivity 17 17 17
   location   4 3 8

   material white
   point      2 1.5 3
   normal     0 0 1

   xdir       1 0 0
   dimensions 4 3

   texname ../images/sky.ppm
   mode 0
}
```

An object oriented implementation in a procedural language

```
typedef struct object_type
{
   char    objname[NAME_LEN];  /* left_wall, center_sphere */
   char    objtype[NAME_LEN];  /* plane, sphere, ...       */
   int     cookie;

/* allow polymorphic behavior .. esoteric objects can provide */
/* their own methods overiding the generics by simply         */
/* overwriting these function pointers                        */

   double  (*hits)(vec_t *base, vec_t *dir, struct object_type *);
                                /* Hits function.              */
   void    (*dumper)(FILE*, struct object_type *);

/* Optional plugins for retrieval of reflectivity */
/* useful for the ever-popular tiled floor        */

   void    (*getamb) (struct object_type *, drgb_t *);
   void    (*getdiff)(struct object_type *, drgb_t *);
   void    (*getspec)(struct object_type *, drgb_t *);


/* Surface reflectivity data */

   material_t *mat;

/* emulate derived classes / inheritance */

   void    *priv;        /* Private type-dependent data */

   vec_t  hitloc;        /* Last hit point              */
   vec_t  normal;        /* Normal at hit point         */
} object_t;

typedef struct plane_type
{
   vec_t   normal;
   vec_t   point;
   double  ndotq;
   void    *priv;     /* Data for specialized plane types  */
}  plane_t;
```

Implementation details

- Detailed course notes made available to the students on a "just in time" basis
- Daily quizzes "encourage" students to review material and ask questions
- Weekly Labs
  - Done right: impose a significant increase in faculty workload
  - Done wrong: a waste of time
- Incremental milestones that are graded are a *must*
- DON"T depend on TA's to be the primary support mechanisms for the students (last semester I responded to over 500 pleas for help by e-mail.)
- DON"T get bogged down in advanced C++ issues.. save something for grad school!!
  - virtual inheritance
  - the diamond ... the procedural plane / the finite plane
  - the augmented diamond
  - templates
  - the standard template labrary

On the not so positive side

Some/most CIS students express NO INTEREST in programming in general and ESPECIALLY no interest in image manipulation algorithms and the mathematical content of the courses. Some become quite adversarial and claim to have been misled by the department.

The "average" student may need more repetition in the development of basic algorithms than is provided in these fast moving courses.

CS-II Assignments

Short programs

sp1 – read in projection data and build a function mapping integer pixel coordinates to world ray direction.

sp2 – driver for model data parser and reflective material definition parser and dumper

sp3 – attribute parsers and dumpers for infinite plane and sphere entities. lookup function for binding entities to materials

sp4 - using the object model presented in class either convert your phase 1 ray tracer from C to C++ or add the sphere object to the sample code provided.

sp5 - build the light_t class to model the point light source entity. It will require a parser, dumper, and a routine that will compute the diffuse illumination at each object hit point.

sp6 – add the finite (bounded) plane of *arbitrary* orientation to the raytracer. It will require a parser, dumper and a "hits" routine.

Labs

lab1 – build library of basic vector functions: length, unit vector, sum, scale, difference, dot product

lab2 – generic linked list function library and use of *make* and *tar*

lab3 – hits function for the infinite plane entity

lab4 – implementation of the *find_closest_object()* function that processes the entity list and identifies the closest object (if any) that is hit by the ray

lab5 – hits function for the sphere entity

lab6 - additional vector functions: cross product, matrix transpose, multiplication of a vector by a matrix, projection of a vector onto a plane

lab7 – conversion of some vector functions to overloaded C++ class methods (not required in the raytracer)

lab8 – implementation of some vector class methods as overloaded operatorts

lab9 – conversion of linked list library to C++

lab10 – addition of internal list element insertion and individual element deletion methods to linked list class

lab11 – a recursive search through a two dimensional maze

lab12 – finding a root of and arbitrary function $f(x) = 0$ using the bisection method (necessary for surfaces of revolution generated by transcendental functions).

Major milestones

mp1 - complete an ambient-lighting only raytracer that supports infinite plane and sphere type objects, Build and submit your own model description file, *mymodel.txt* that demonstrates your program works correctly in a technically interesting and artistic way.

mp2 – the final raytracer

Ambient lighting with planes & spheres	45
Diffuse lighting / shadows	10
Finite plane	5
Textured planes	10
Spotlights	5
Projecting textures	5
Specular reflection	5
Specular glints	5
Anti-aliasing (makefile configurable)	5
Tiled infinite planes	5
Multiple non-quadric revsurfs	10

Possible	110

Codelab

- A spinoff of an NSF funded project of the SUNY (CCNY?) system.
- Provides an easy way to enforce repetition thus helping the "average" student.
- May irritate the above average student.
- The proprietors are quite helpful and knowledgeable in CS ED
- *Particularly useful in first course in a new language.*

Texnh-II deliverables (EAE)

Evaluation, adoption, extension

Evaluation

Continued application of existing and new assessment mechanism

Adoption

DON"T create a cast-in-stone curriculum
DO provide faculty mentoring
DO encourage innovation and evolution

UNC-W / WCU
Will teach at least one course in each of two years using texnh approach

UNC-W a CS-1 type course
WCU a CS 215 type course

Extension

Clemson will extend the texnh approach to senior divsion courses.. possibilities include

Parallel processing: radiosity modeling on the GPU.
Operating systems: implementation of a graphics or audio device driver.
Information and coding theory: analysis, encoding, and decoding of audio streams.
Database management systems: design and implementation of a multimedia database.

Collaboration plan

This meeting
Clemson PI's present colloquia at UNC-W and WCU
Meeting next summer primarily to discuss and refine assessment mechanism

Deliverables

Each partner will submit a final report describing their experiences. The report will be in a format suitiable for submission to SIGSCE or similar conference.

The external evaluator working in conjunction with the CPATH evaluator will assess our assessement