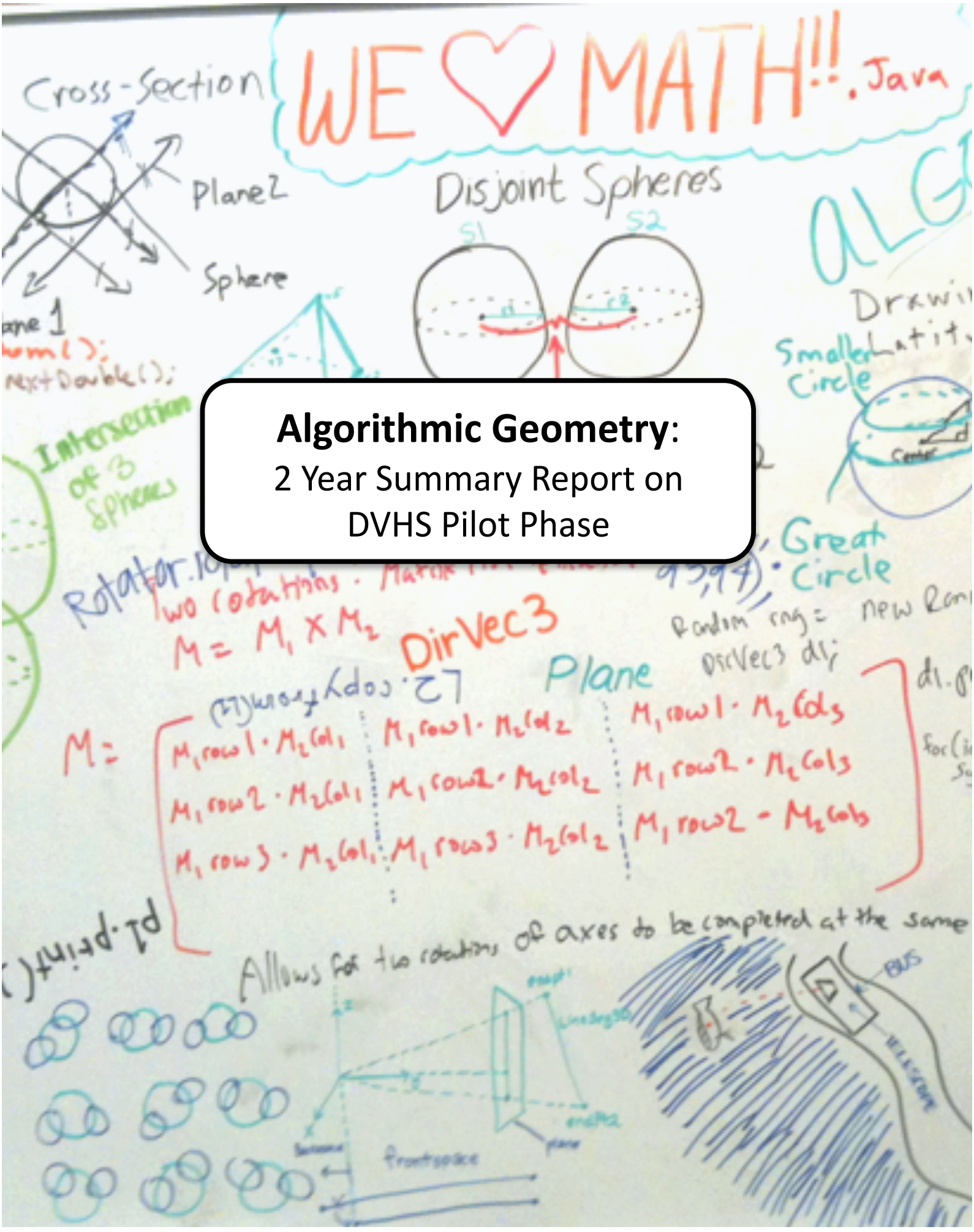


Algorithmic Geometry:
2 Year Summary Report on
DVHS Pilot Phase

Algorithmic Geometry: 2 Year Summary Report
Dougherty Valley High School Pilot Phase

We're on the move

Everyone involved with Algorithmic Geometry math education for high school is advancing significant Math content modernization. Our BHAG: Can teens learn geometry problem-solving the way it has evolved in our tech industries, as a *seamless integration of math and computer science disciplines*?

The payoff for students is up-to-date, hands-on preparation as inventive problem-solvers, and the confidence that comes from tackling wicked difficult spatial problems through cleverness harnessing prodigious amounts of software computation. Learning the ropes used by today's math-savvy tech professionals, students gain admission to an *insider's* understanding of 21st century apps such as 3D graphics, computer vision, GPS positioning and robotics. Algorithmic math is an essential adaptation needed to keep 9-12 math *relevant* in a digital world.

Keeping college-prep math content *current* with the STEM professions is *a 9-12 leadership responsibility*. Your support and interest in this research project puts you in that vanguard. It's a messy process, but together, we're on the move!

Demonstrating the Possibilities

The major project achievements over the past 2 years at Dougherty Valley HS (San Ramon CA) were:

1) **Content development** – We mapped the learning sequence in *Flexing the Power of Algorithmic Geometry* onto the 145 hour class schedule. A spectrum of PBL challenges were developed, ranging from 20 minute mini-challenges to a 2 week "software company" class project. We prototyped flipped mini-lectures delivered over the web. Public speaking was embedded as a core competency expected of all innovative math thinkers.

2) **Assessment development** – We prototyped and refined a set of formative online quizzes that reinforce and stretch the lectures and labs covered in class. For summative assessment, we developed two semester Final Exams that measure *operational* problem-solving knowhow in a novel situation designing & writing software – *algorithmic math inventiveness*.

3) **Student success baseline** – For n = 28 students, we have initial results on what student success with this type of project-based, advanced math-CS learning looks like.

4) **Logistics practicum** – We obtained a 1st cut answer to the question, "What does it take to successfully implement such a course in public school?" We have preliminary answers in regard to professional development, team-teaching model, course pacing, online learning supports, grading policy, software lab tech support, student recruiting and master scheduling.

copyright Spatial Thoughtware 2012

5) **Outreach / Alliance**

- Algorithmic Geometry approved as "c" advanced math by UCOP 2010-12
- Lawrence Livermore National Labs contributed funding and ideas
- DVHS Academic Booster Club (ABC) donated funds
- Press and local cable TV covered the pilot course
- Visit by Jerry McNerney, Member of Congress w/ PhD in Mathematics
- Parents night open-house attendance was strong both years (~70%)
- Students presented projects both years at the summer Computer Science Teachers Association CS4HS conference at UC Berkeley (sponsored by Google)

Project-Based-Learning (PBL) Overview

The big conundrum facing math ed is how to keep from *turning off* students. One prevalent student complaint is "When am I ever going to use this stuff (except on tests)?" The antidote, a pillar of the Common Core Math Standards, is to infuse problem-solving with real-world applications (recontextualizing math).

To fit PBL into this math course, the theory content was already *whittled down to minimalist essentials*, just enough to undergird the application challenges. A fundamental advantage of algorithmic math regarding PBL is the ability of students to incrementally vest their math knowhow (as it is expanding) in the medium of software. It is difficult to appreciate the time-efficiencies implied by this workstyle until you have personally experienced the remarkable mental leverage of having all your past mental problem-solving work stored in the form of finished, reusable algorithms. Project synergy and efficiency are the result.

Typical project challenges are designed to be solvable with 2-5 hours of effort, broken down in approximate fourths (new math theory, new sketches, new Java code, new computer graphics). Though the student may only write 30-50 new lines of code for the project, when this code runs, it is exercising *90% of the software code and past mental work the student has expended up to this point in the course*.

This pattern of incremental, general-purpose knowledge accretion is the virtuous cycle that empowers algorithmic math learners to surmount problems too complex for traditional (paper and pencil) math analysis. The complexity resides in the ~100 hours of problem-solving effort, crystalized in the student's library of software algorithms, and its ability to be quickly *repurposed* to solve an unexpected new problem. In the student's estimation of effort, the complex project takes 2-5 hours to solve -- the 100 hours of previous work *it depends upon* is discounted as *unrelated* problem-solving. *Taking for granted* their algorithms already solved, students experience the power of solving a wicked-complex problem with minimal time and effort. This economy and fluidity of math brainpower is *entirely new in the past 30 years* thanks to high-level programming languages. The amplification of mental power that algorithmic thinking brings is a *game changer*, and will come to differentiate 21st century math from that of previous centuries.

Our experience and data collection indicate that students are *turned on* to their own creative

genius and the relevance of mathematics when thrust into the role of high-tech problem-solvers!

PBL Example: "Plane Down" (details in Appendix A)

This report would be incomplete without conveying at least one example, start to finish in all its gory details, of how PBL and algorithmic geometry engage the student's thirst to conquer real-world situations.

"Plane Down" takes students back to the overnight disappearance of Air France 447 en route from Rio de Janeiro → Paris June 1, 2009. Students have to determine where to mount a search and rescue operation (longitude, latitude), given only the following information available the night of the incident:

Rio airport [longitude, latitude]	Scheduled flight duration
Paris airport [longitude, latitude]	Flight duration up to loss of radio contact

The implied task is to calculate the flight path (a great circle arc), then interpolate along it proportionally based on flight duration. The motivational hook is the *sense of urgency* in rescuing people bobbing in the ocean. Problems are chosen to be equally compelling for boys and girls.

From a Common Core Math perspective, this problem naturally builds in these *complicators*:

- Geo-locations given in spherical angles (phi, theta)
- Flight path (a tilted 3D circle) benefits from using 3D *direction vector* geometry
- Overlapping coordinate systems: Geo-locations vs. Cartesian coordinates [x y z]
- Mixed units: degrees, radians, kilometers
- Cartesian origin is implied, not specified
- Novel math challenge: How to interpolate locations along a tilted 3D circle?

This challenge took 4 class days (3.3 hours). All students (100%) solved it with their own software code and numerical results, working collaboratively in pairs. Two distinct algorithmic solutions to 3D circular-arc interpolation emerged. No instructor prompting was given on this sub-problem, nor was there anything helpful found on the web. See **Appendix A** for the details of "Plane Down" and an example student solution.

Summative Assessment – 2 year Final Exam Results

Our assessment philosophy: *If students are not able to apply math thinking in a novel situation, then it's dubious what has been learned*.

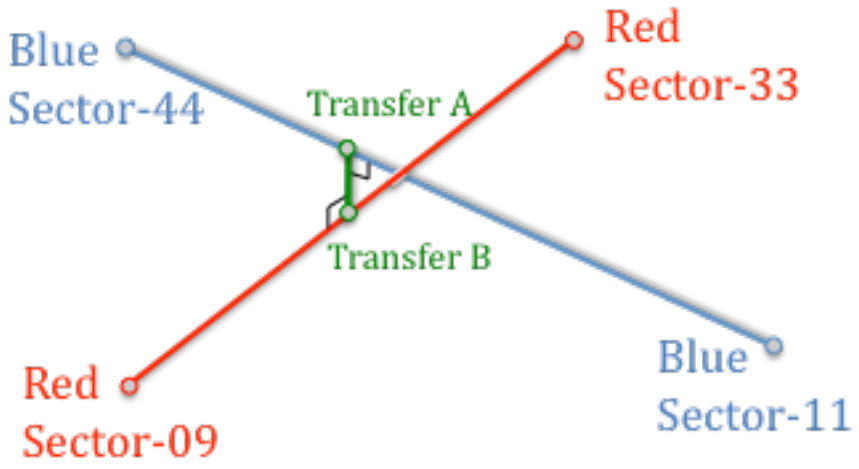
Mindful of students' highly advanced skills in memorization and regurgitation, all AlgoGeom exam items oblige some degree of novel thinking. All Final Exam questions involve 3D geometry (1st semester exam covers 2D). The exam is open book, notes and Java source code. The web is available, but not very helpful.

Observing a 2.0 hour test duration, the exam is structured to probe these 4 areas of proficiency:

Math-fact short-answer (6 items). These measure the student's ability to reason using vector math, 3D geometry, and Java coding fundamentals, for example reasoning how the vector dot product (and cross product) output behaves when the input vectors **u** and **v** are perpendicular.

Representational creativity (single 3-part item). During the course, students learn how to represent these 3D objects parametrically: 3D Point, Line Segment, 3D Direction, Sphere, Plane, 3D Circle, and 3D Extended Line. In this item, students are shown a novel object (a Torus), and asked to create a parametric representation (set of variables) capable of representing all possible Tori. This item demands students think like professional mathematicians.

Application Utility (single 2-part item). During the course, students write ~200 algorithms. A key skill in algorithmic problem-solving is recognizing when an algorithm you've already solved may be used to advantage in a new context. In this item, a futuristic scenario is given involving two Vertical Cities "people-mover" linear tubes (think ultrafast escalators).



As architect, the test-taker has to create the shortest transfer tube (Green Line) linking up the Red and Blue lines, solving for 3D coordinates of its endpoints and the tube length. The givens are the coordinates of the 4 end-of-the-line stations.

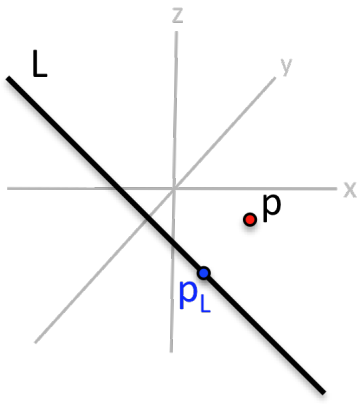
Only one of the ~ 200 algorithms solves this application, *the closest approach Line Segment of 2 skew lines*. (In 2D geometry, the analogous problem is intersection of 2 lines → 1 point). In addition to selecting the right algorithm, the student must write a short amount of Java code to enter the givens, run the algorithm on them, and output the answers (run a Test Case).

Algorithm Design (single item - inventive sketching, Java coding and running Test Case). This item will consume between 0:30 and 1:30 of the exam time. This is the only long-response item. It assesses the paramount skill developed during the course -- creatively solving a novel math problem algorithmically. The problem given (Prob 9) is:

Challenge: Algorithm Design, Implementation, and Test Case

9. Design an algorithm to solve the following general problem:

Given an extended line **L**, and a point **p**, compute the point $\mathbf{p}_L$ on the line closest to **p**.



a) ***The Design Data Sheet (sketch) must be completed to receive any credit.***

b) Test Case (attempt only after algorithm is designed and coded)

Line **L** passes through these 2 points: [-35 -67 99] [22 -72 -105]

Point **p**: [13 13 13]

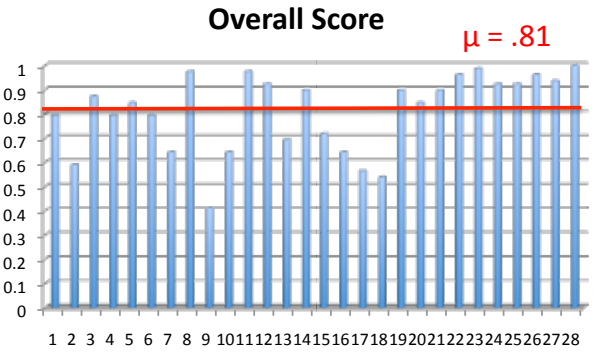
$\mathbf{p}_L$ = [_____ _____ _____]

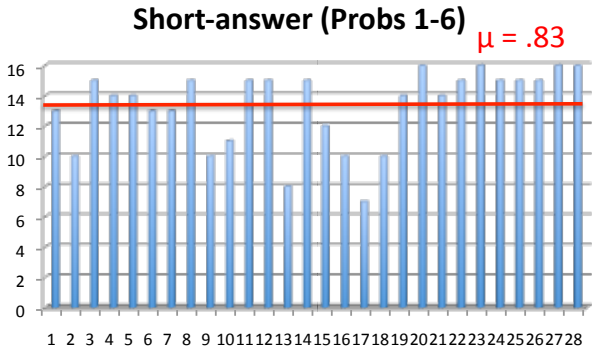
c) How could you verify that the path from $\mathbf{p}_L$ to **p** is orthogonal to line **L**?

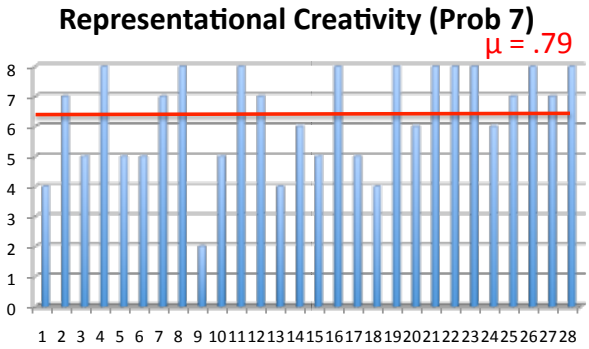
_____

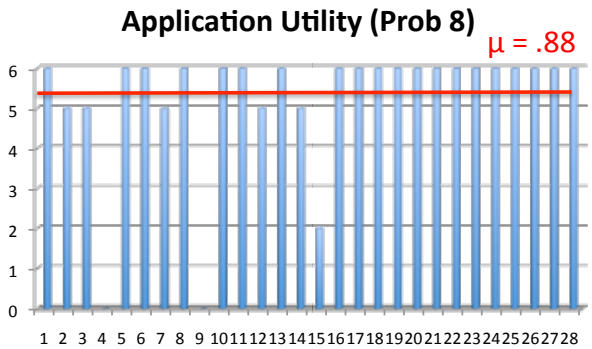
Now run this check. What was the result? _____

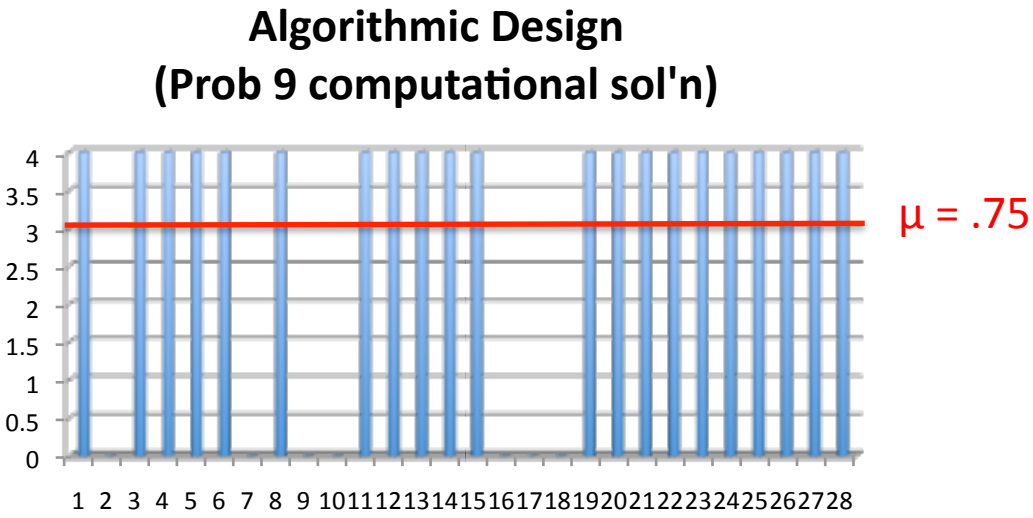
2-Year AlgoGeom 3D Final Exam Results for n=28 students (12 female, 16 male)











The Algorithm Design (Prob 9) is scored P/F on whether the student was able to solve the test case numerically (which requires a correct algo to have been designed, implemented and test case set up). 21 / 28 students (75%) were successful at this high level of inventive proficiency under an exam deadline. Success correlated highly with sketch quality.

Three distinct algorithmic approaches were invented by students, evidence for creative juices flowing on this complex task. Yet, the scoring is entirely objective – other than sketch quality (scored manually by the instructors), the algorithm either works to obtain the correct answer or not. The odds of obtaining the correct numerical result from an incorrect algorithm are vanishingly small, as are the odds of a correct algorithm absent a sketch outlining it. Students accept algorithmic testing as an impartial and objective yardstick of success.

Significance of Learning Achievement Results

The Project-Based Learning results are presented alongside the Final Exam results as two parallel indices of student learning achievement. When doing PBL, task time is elastic and accomodates to the last problem-solvers ("group master then advance"), so individual speed is downplayed. PBL draws upon collaborative design (pairs or threesomes) and individual implementations, rehearsing typical roles in the tech workplace.

The Final Exam is a purely individual performance task of both algorithm design and implementation inside a rigid timeframe. In order to keep our measurements of *problem-solving ability* unadulterated by *performance speed*, we give students 4X the time it takes the instructor to take the exam (i.e. items are pared down until the instructor takes <= 0:30 min). The "Overall" Final Exam score distribution looks unremarkable (mean = 81 / 100), and no different from any other advanced math course taught at the same school. We take this as evidence that AlgoGeom content in *no harder to learn* than other 11-12 grade math courses (Calculus, Statistics, Discrete Math).

Considering the *advanced nature* and *unrehearsed novelty* of the PBL and Final Exam problems, the project has obtained encouraging preliminary evidence that the Algo Geom course is effective at imparting a level of spatial problem-solving sophistication well beyond what most educators would think possible for high school. Our mission is to raise those expectations.

Demonstrating the "quantum leap" in problem-solving complexity offered by interdisciplinary Math-CS is what we had planned to achieve with these DVHS pilot courses. The starting point was establishing feasibility of the course content and the learned proficiencies obtained with it. In this regard, the project has scored a "hit", and made it to first base.

What Did it Take?

The project learned some things about what it takes to begin offering Algorithmic Geometry in a public school setting. This discussion is based on 1 data point – DVHS and San Ramon Valley USD -- it is premature to extrapolate from it. The point of this section is to document the strategic decisions, course preparation and deployment thinking that went into staging the pilot courses.

Initial district pitch / NSF proposal: Three years before launch, the course developer (Spatial Thoughtware) met with the San Ramon Valley USD Math leadership team (Stan Hitomi, Gregory Duran et al.) to explore options for Algorithmic Geometry. One year later, a NSF grant proposal

(DRK12) was jointly submitted which was not awarded. The following year, a U.S. Dept. of Education (Math and Science) proposal was jointly submitted.

DoEd proposal / School Board Approval: While the U.S. DoEd grant was being reviewed, the District decided to launch Fall 2010 regardless of the grant decision. Board approval came 6 months prior to launch. Lawrence Livermore Labs funding was arranged.

Professional Development tutoring: Two credentialed 9-12 Math teachers were trained as AlgoGeom problem-solvers in 1:1 tutoring workshops (both beginners at Java programming, but familiar with vector math theory). This workshop ran 7 months leading up to the course launch, a mixture of p2p tutoring and remote GoToMeeting sessions. The number of hours was approximately 80 h /per teacher. Beth Injasoulian (1st teacher trained), our Professional Development expert, made substantial contributions to the 3D AlgoGeom semester module, though she has yet to teach a class. Gregory Duran, the 2nd teacher trained, co-instructed the DVHS course 2010-2012, and contributed extensively toward courseware refinement.

UCOP Approval: During the 2010-11 school year, DVHS applied to UC Office of the President to obtain a-g vetting as a "c" Advanced Mathematics course (countable towards the UC/CSU admission requirements). It took 3 submissions before being approved in March 2011.

Paired-instructor model: We adopted a team-teaching model, pairing an AlgoGeom-trained, credentialed, teacher-of-record (Duran) with the course developer (Bierre), a computer scientist bringing AlgoGeom and career Java expertise. The lecture load was split, as was the lab mentoring. Some online mini-lectures were created (Duran).

Online portal with formative quizzes: A course online portal was created (Lukoff). Chapter quizzes, all administered and scored online were provided as a way for instructors to monitor individual progress while boosting student confidence. Unlimited quiz retaking was allowed up until a deadline, and quizzes counted minimally toward the grade, in order minimize quiz anxiety. 22 / 28 students worked to complete all quizzes 100% correct. Students could also pull down lecture Powerpoints, lab Java code snippets, and project files. The portal became important as a dependable way to share code.

iMac Lab: The Java can be done on Mac or Windows. We opted for an iMac lab, where each student had a dedicated computer. Students used Eclipse / Java / Java3D (mainstream professional tools) as the development platform, along with browsers and PowerPoint. A large-screen projector coupled to the teacher's rig, a doc-cam, and whiteboard were used as a shared workspace. Each student kept a lab notebook of sketches (paper or web-based).

Textbook: *Flexing the Power of Algorithmic Geometry* (Bierre) served as the textbook (syllabus plan). The school district purchased books bundled with AlgoGeom software seats and teacher training. The book was not used in class, but helped students who missed class.

Group-master-then-advance. Syllabus pacing was flexible with the entire group reaching mastery before moving forward. Extensive reuse of previous algorithms dictates that the class stay together in formation while ascending the learning curve. Absenteeism was light, and was

dealt with by borrowing code from a student peer. The psychological impact of group-master-then-advance is *learner safety* – students know they are not going to be allowed to fall behind. To prevent boredom on the part of the fast learners, *stretch problems* were made for most labs.

Grading policy. 70% lab completions, 10% final exams, 10% online quizzes, 10% class participation and public speaking. Homework was rare, respecting the need to have tech support 100% present during early Java programming learning. Our strategy was to avoid a high-stakes exam situation; AlgoGeom was positioned as an advanced math course taken for enjoyment and intellectual growth.

Semester Final Exams. Exam designs were a team effort, under the close supervision of our Assessment expert (Lukoff). A master list of proficiency goals drove test item design. QA for exams consisted of instructor administrations, allowing a 25% time-target.

Software lab tech support. AlgoGeom students spend >50% class time developing software. The computer lab is a critical daily asset, and needs tech support staffing. In addition, *course instructors* (and TAs / Automated Mentor in the future) need sufficient understanding of the labs and Java programming to rescue any student who has become stranded.

Student recruiting. As an elective *not eligible for grade inflation*, AlgoGeom had to compete for signups unfairly against AP and Honors PreCalculus courses that inflate GPA. This was a major impediment to fulfilling a 25-signup school requirement. Counseling told us we could fill 2 sections if it were an Honors course. UCOP limits each subject area to only 1 Honors course. This policy works against expansion of rigorous course opportunities.

Master scheduling. AlgoGeom is a 2-semester course (or 3 trimester). Because programming labs are hard to finish at the bell, the best schedule slots are periods that *precede brunch or lunch*. The advantage of the paired-instructor model is that the Java expert can remain after class debugging student programs where needed. A note is simply typed into the student's source code giving mentoring on the root cause of the problem. This system of debug support can likely be done remotely in the future.

Financial support. The school had to raise funds to offset the partially filled class sizes. We thank Lawrence Livermore National Labs for a $10K , and the DVHS Academic Booster Club for $8K. The lab equipment and textbooks were covered out of existing budgets. The Eclipse and Java software developer tools are open-source (free). The course developer (Spatial Thoughtware) charged ~ $110 per student for textbook and courseware.

Teacher, Parent and Administration support. DVHS was an ideal site for piloting AlgoGeom, due to many factors: The school educates the children of a Silicon Valley tech workforce; 2 key district decisionmakers (Gregory Duran, Stan Hitomi) are tech industry veterans. We had support for doing this course from the School Board and Superintendant, Principal, Math Department and IT staff. We held two events for parents, and found them extremely supportive of Math education laced with Computer Science. We received favorable attention from the press and our Congressman, Jerry McNerney, a PhD in Mathematics.

2012 NSF Computing Education proposal. We jointly proposed to this new program at NSF, presenting one year of public school pilot experience with a plan to grow out 2 new schools. All the 2012 funds ended up going to establishing *core* Computer Science education programs. Interdisciplinary CS courses were invited in the RFP, but were deprioritized after an unexpected program budget cut of 50%.

Now for the *hard question*: What will it take to expand access to Algorithmic Geometry education? In the public school sector, the major obstacles to scale up are filling full classrooms, and developing teachers. Math content modernization could be ushered in under Common Core (although the CC Math Standards somehow failed to embrace *the major trend* reshaping academic and industry math practice: *software programming*).

While teaching the pilots, we constantly asked the question: "What are we doing in the classroom that could not be done in a *web-based* AlgoGeom course?" With two years under our belt, the answers are clear: The social and motivational aspect, what Vygotsky calls *social induction,* works best in a p2p setting. We don't see a practical way to teach *sketching* over the web using a software tutor. (Joe LaViola's work in pen-based tutoring might be relevant in the future). The debugging of student labs (keeping the beginner programmers moving forward) also calls for expert human interaction and judgment. At this point, skilled human teachers (Math/CS) seem an indispensible ingredient of the learning results.

The good news is that we surmounted the earliest professional development challenge, getting a couple of Math teachers operational with the AlgoGeom/Java problem-solving methodology. Granted, both were highly motivated, seasoned, expert Math teachers, with tech industry experience. This pedigree defines the *ecology* for expansion of Algorithmic Geometry education – there is no suggestion than *every* 9-12 Math teacher could be trained. We would only need to impact about 5% of 9-12 Math-CS teachers. This could develop ~100K students per year, enough to have a broad impact on the future *inventive workforce*.

Finally, the process for getting laboratory Math-CS started in a public school is complex and time-consuming. There isn't any roadmap or process, nor is this topic on anyone's Master Plan. The federal grant programs are not structured to support interdisciplinary Math-Computer Science. The 2-year DVHS AlgoGeom pilot owes its existence to *bottom-up leadership* – a half-dozen key individuals saw it as their responsibilty to begin reforming public education to more effectively prepare young people for the software tech economy they are inheriting.

And, another dozen *advancers* supported AlgoGeom at key junctures to make it happen (see Appendix B: Project Contributors and Advancers).

The hiatus in teaching after 2 years is a good opportunity to stop and reflect, gather our wits and plan the next move. In some fashion, the project's fate lies in the hands of people with access to funds, and institutional power, who are *as passionate* about reinventing Math Education as all of you are!

Pierre Bierre, Project Director, AlgoGeom.org
August 2012

Appendix A: "Plane Down" example of Project Based Learning

Problem definition handout:

Air France flight 447: Rio de Janeiro – Paris

On June 1 2009, Air France 447 crashed in the Atlantic Ocean during a storm.

The flight was scheduled to take 11.1167 hr.

However, radio contact was lost 4.1833 hr into the flight.

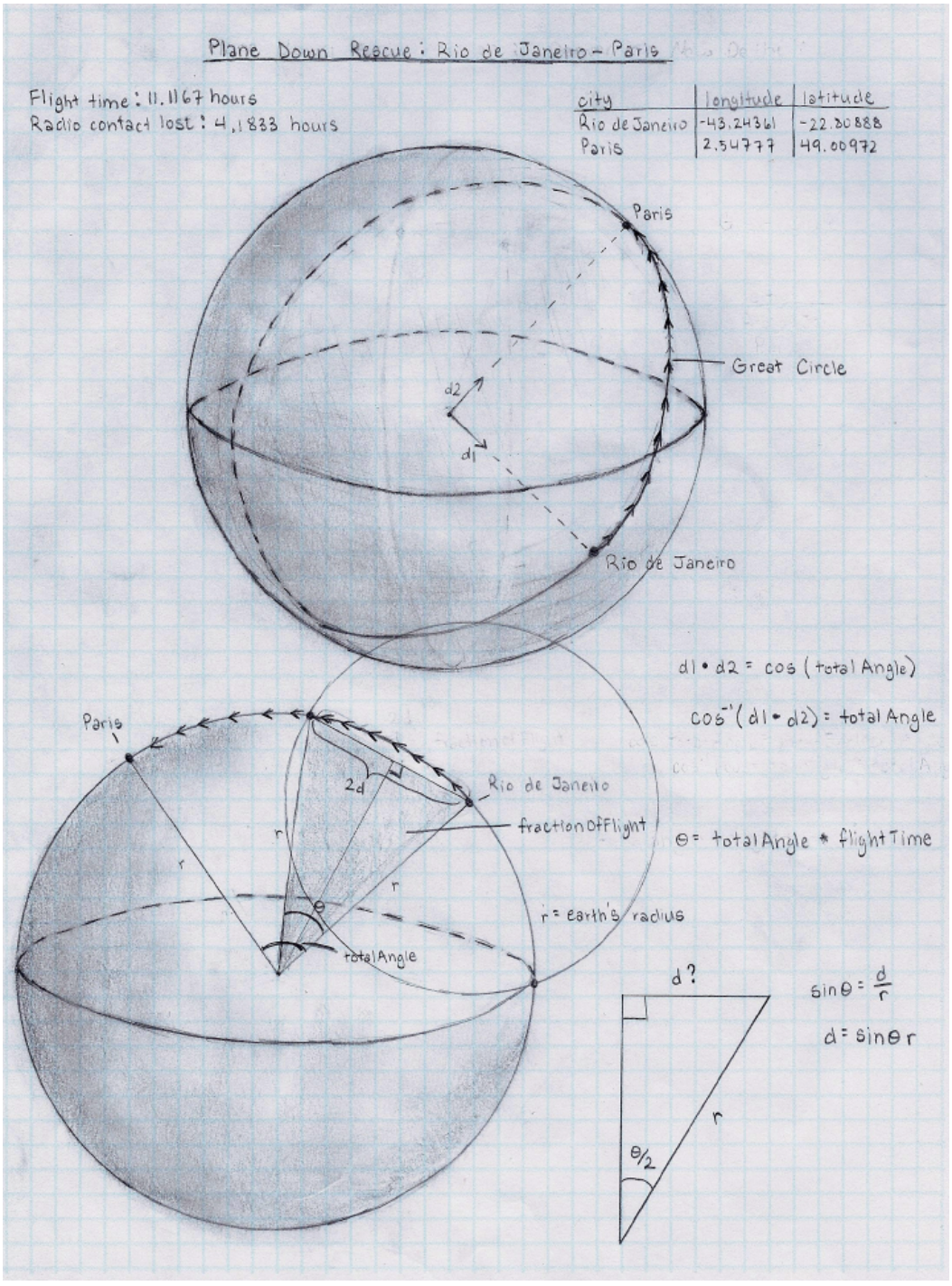
Assume the plane flew at normal speed right up until radio silence. Your job is to pinpoint

[longitude, latitude]

for the search and rescue mission.

city	Longitude	Latitude
Rio de Janeiro	-43.24361	-22.80888
Paris	2.54777	49.00972

Student sketch outlining solution steps:



The sketching phase is the inductive, creative workspace where the solution first emerges. Quality sketching is an *inventor-level skill* that must be developed to be successful in Algorithmic Geometry. This is because the problem-solution will only "jump out" from the sketch when sufficient visual cues are present in it to gang up on, and fire the neurons in the brain representing the solution….the "aha" moment.

Student Java source code:

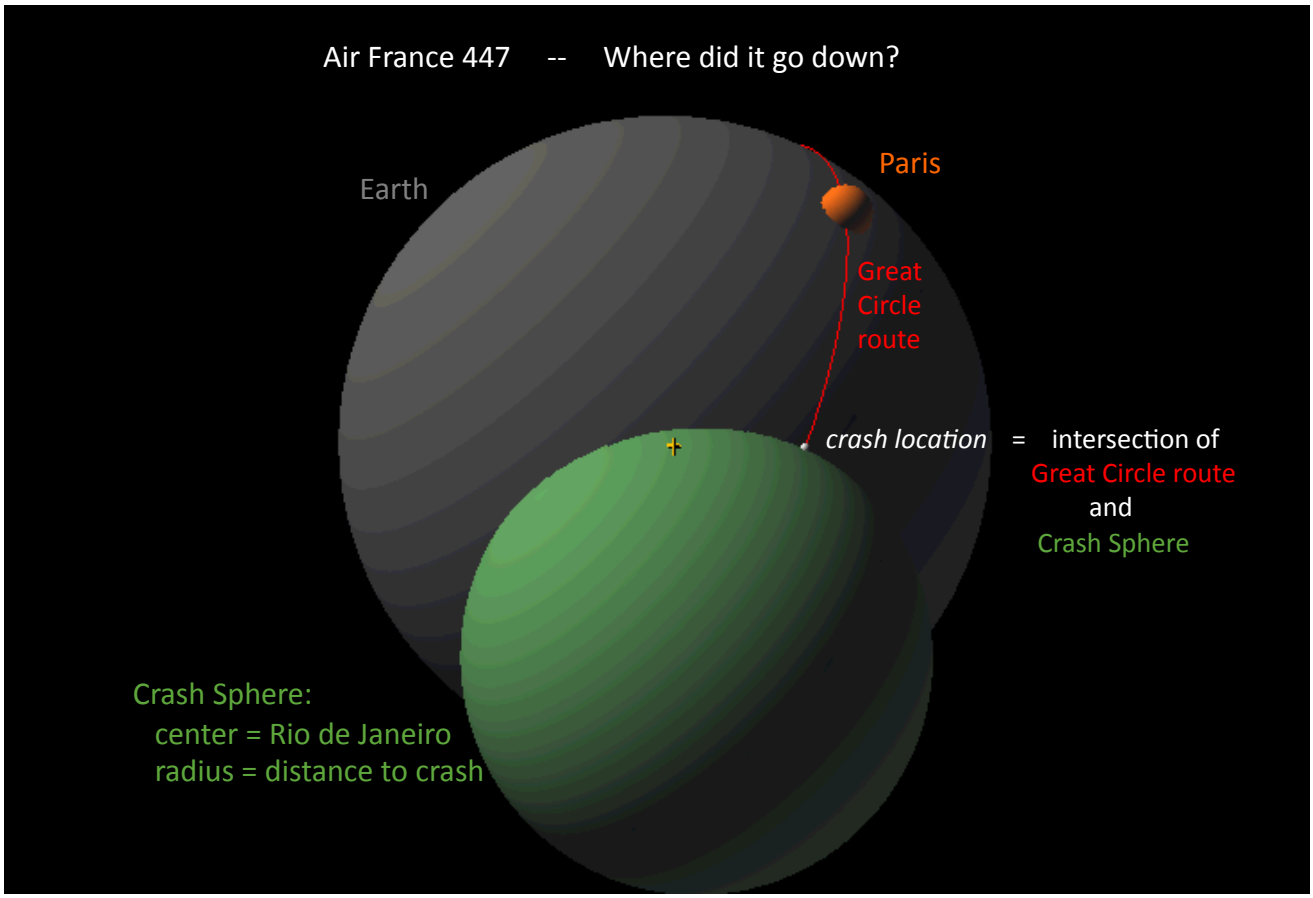
```
// ********************** airFrance447PlaneDown **********************
private void airFrance447PlaneDown() {
    //convert [longitude, latitude ] to geocentric DirVec3
    DirVec3 dRio = new DirVec3 (radiansOf(-43.24361), radiansOf(-22.80888));
    DirVec3 dParis = new DirVec3 (radiansOf (2.54777), radiansOf(49.00972));
    //compute airport locations in 3D cartesian coords
    Vec3 pRio = dRio.scalarMult(20.0/Math.PI); //1:1000 scale model  earth radius =  40000km / 2pi
    Vec3 pParis = dParis.scalarMult(20.0/Math.PI);

    Sphere earth = new Sphere (new Vec3(0,0,0), (20.0/Math.PI -0.02));
    contentModel.addToModel(earth.renderable(), new Color(80,80,80)); //32,150,150 <--teal

    Sphere Rio = new Sphere (pRio, 0.5);
    contentModel.addToModel(Rio.renderable(), new Color(255,255,0));

    Sphere Paris = new Sphere (pParis, 0.5);
    contentModel.addToModel(Paris.renderable(), new Color(255,100,0));

    Circle3D greatCirRoute  = new Circle3D (new Vec3(0,0,0), DirVec3.crossProd(dRio, dParis), 20.0/Math.PI);
    contentModel.addToModel(greatCirRoute.renderable(), new Color (255,0,0));

    double cosAngle = DirVec3.cosineOfEnclosedAngle(dRio, dParis);// arc of planned flight
    double thetaRadiansRioParis = Math.acos(cosAngle);
    double thetaRadiansRioCrash = 4.1833 * thetaRadiansRioParis / 11.1167; // arc of partial flight
    System.out.println(thetaRadiansRioCrash);

    double distCrash447 = 2 * greatCirRoute.r * Math.sin(thetaRadiansRioCrash / 2);


    Sphere crashBubble447 = new Sphere(pRio, distCrash447);
    contentModel.addToModel(crashBubble447.renderable(), new Color (80,150,80));

    TwoPoints3D intPts = Sphere.intersectionOf(greatCirRoute, crashBubble447);

    //1st solution point is crash location; plot, then convert to [long. lat ]
    Sphere crashSite = new Sphere(intPts.p1, 0.1);
    contentModel.addToModel(crashSite.renderable(), new Color (255,255,255));

    DirVec3 d_int1 = new DirVec3 (intPts.p1);
    Vec2 crashSite1rad = d_int1.toAngles();
    Vec2 crashSite1deg = new Vec2 (-360.0+degreesOf(crashSite1rad.x), degreesOf(crashSite1rad.y));
    crashSite1deg.print();
}
```

This source code both generates a 3D graphics model of the problem & solution, and calculates numbers [longitude, latitude] for the search and rescue location.

Note the data types (DirVec3, Vec3, Sphere, Circle3D, TwoPoints3D) used to define the high-level variables of the problem. These data types were developed by every student earlier in the course, and are usable across a wide swath of 3D problems.

The graphics generated are shown on the next page.

Student 3D graphics of the problem and solution (annotations added):



The creative solution to the problem can be seen in the student's 3D graphic model.

First, 3D locations [x y z] are determined for the Rio and Paris airports (the dot for Rio is inside the Crash Sphere at its center) . Next, the Great Circle route is constructed, passing through the two airport locations, and centered at the Earth center.

How far is the crash from Rio in km (straight line 3D distance)? The student solves this problem in the sketch using the trigonometry of chords and arcs. Knowing this distance allows the student to "bubble out" from Rio, i.e. construct a sphere with this precise crash distance as its radius….the crash location must fall somewhere on this green Crash Sphere.

The crash location is where this Crash Sphere intersects the Great Circle route flown by AF447.

How did this solution occur to the student? It has a lot to do with the student knowing all their tools – in this case, *intersection of sphere and 3D circle* had been solved about 1 month prior to the "Plane Down" assignment. The computed crash site [longitude, latitude] result:

```
[ -29.021125909848536  4.9882188783061245 ]
```

What are the take-homes from this example of PBL? This is just a starting list of the most important ones:

Ownership of Learning. The instructors give out the challenge, and then get out of the way. The kids come in and immediately settle down on-task every day. Though it will take longitudinal tracking of student outcomes to measure it, we believe math education will emerge *stickier* as teachers shift more learning responsibility to students via PBL.

Nurturing invention. Where do new technical breakthroughs come from? How can we develop inventive originality and confidence? With so much pedantry on the www a few clicks away, it is actually becoming harder to instill the patience, focus and self-confidence needed to craft inventions from first principles. The temptation to "Google it" can be overwhelming. Many students starting AlgoGeom bring a negative orientation toward sketching, which has to be turned around before the student can function as an inventive geometry problem-solver. The computer is such a powerful distractor, we moved some of the theory-sketching activities out of computer lab. On the final exam, we give a design problem that assesses inventive algorithmic problem-solving.

Advanced 3D geometry. The problems we tackle in the course are college-level, some pushing the advanced degree envelope. What is striking are the *dozens* of such problems our students vanquish in a 145 hour course. What explains this productivity? The AlgoGeom course cohesion comes from the fact that *software-implemented vector math* provides a *common foundation* for all the spatial applications that are transforming our world. Students who can write a vector geometry library in software and use it effectively in graphics space, by definition, are prepared for advanced problem-solving in a myriad of scientific and engineering domains. The generality of Algorithmic Geometry qualifies it as a prototype for a future standard Advanced Math course. If we want this knowhow disseminated broadly, it must be introduced in high school…college is too late to have a broad impact.

Making math come alive. Selling math to young people can be easy or hard. Presenting it as abstract, decontextualized, and chock full of dense notation is the hard way. The essential insight for math course developers is that human understanding proceeds from the concrete to the abstract. Algorithmic Geometry exploits operant (hands-on) learning of vector math, using student-written 2D computer graphics to visually experience the effects of vector addition, scalar multiplication and normalization. The use of graphic simulation (2D and 3D) under student programmatic control is the *ideal medium* for learning spatial mathematics and its applications. The abstractions develop naturally as patterns in problem-solving beg to be given names and shorthand notations. This *student-centric* rollout of knowledge works toward eliminating math anxiety. And, when the student's graphics come alive on the screen, the math theory driving them instantly gains credibility and relevance.

Collaboration & communication. When students work in small groups, the chatter is on-task as students verbalize their nascent understanding of geometric programming. Rehearsing and delivering stand-up presentations, students show dramatic improvement.

Appendix B Project Contributors and Advancers

AlgoGeom.org (Spatial Thoughtware)

Pierre Bierre, Project Director, Algorithmic Geometry content creator, software developer, 2010-2012 DVHS course co-instructor, AlgoGeom website developer
Gregory Duran, Math Education Lead, DVHS / SRVUSD partner lead, 2010-2012 DVHS course co-instructor
Brian Lukoff, Chief Education Scientist, Assessment Lead, online portal developer
Beth Injasoulian, Professional Development Lead, AlgoGeom3D tester
Cliff Braun Cal Poly student, educational robot arm co-developer

San Ramon Valley Unified School District

Stan Hitomi, District Math-Science Coordinator
Christine Williams, Assistant Superintendant
Rob Stockberger, Director, Secondary Education
Toni Taylor, Director, Categorical Programs
Marie Morgan, Director, Instruction and Staff Development
Terry Koehne, Director, Community Relations
Sean Davis, IT Computer Lab Technician
Dr. Steve Enoch, Superintendant (retired)

Dougherty Valley High School

Gregory Duran, founding Math Department Coordinator
Denise Hibbard, founding Principal 2006-11
Dr. Jason Reimann, Principal, 2011-
Jessica Coulsen, UCOP "a-g" Representative
Nancy Walters, IT Coordinator
Academic Boosters Club, **Anupam Rastogi**, 2010-11 chair

Lawrence Livermore National Laboratory

Richard Farnsworth, Director, Teller Education Center
James Bono, Director, Public Affairs

U.S. Congressional District CA-11

Hon. Jerry McNerney, House of Representatives
Nicole Alioto, District Manager

Computer Science Teachers Association

Chris Stephenson, CSTA National Executive Director
Dan Garcia, UC Berkeley EECS
Eugene Lemon, CSTA Golden Gate Chapter Pres.

US Govt. Grant Program Directors

Dr. Christina Chhin, U.S. Dept. of Education, NCER Mathematics and Science
Dr. Jan Cuny, National Science Foundation, Computing Education in the 21st Century
Dr. Spud Bradley, National Science Foundation, Discovery Reseach K-12

[Cover art: A study guide mural created by the 2010-11 AlgoGeom students]

generate point on line (double y_pos)

Xaxis = [1,0,0]

Yaxis = [0,1,0] [a,v,1]

Zaxis = [0,0,1]

intersection of [g] plane and sphere

x [1,0,0]

concentric circles?

ScalarMath.equal()

CW Rotator(newx, newy)

intpt2

intersection of circle and line

intpt1

concentric

SPH1 SPH2

System.out.println(Vec3);

cross product

DirVec

x = a

y = c

z = s

disjoint

Cir

SPH1 SPH2

a × b = c

C = ((a.y * b.z) - (a.z * b.y)

a

b

c

Lean

intpt1

for(int i=0; i<100; i++){

degrees = 360

cir1 }

radians = 2π

Midpoint-3D:

((a.x+b.x)/2, (a.y+b.y)/2, (a.z+

cir2

intpt2

intersection of 2 circles