Exam QA

- 5 hours, all written materials + calculator.
- I prefer C++. Exam by hand. Dont save the trees.
- We are doing math - so make sure all relevant math related code is implemented - you can ignore or gloss over windows, drawing etc.
- Explain what you are doing - but dont explain too much.
- I will come by and I will answer all questions which do not have influence on your grade.
- 50/50 math/programming. My goal to have something from each module.
- Well prepared if you understand hand-ins + homework tasks + lectures and you should be well prepared for a C.
- Score 0-100.
  - $>$ 40 E
  - $>$ 50 D
  - $>$ 60 C - congratulations
  - $>$ 80 B
  - $>$ 90 A

Exam QA

- I will put a list tomorrow of people who are ready for the exam on fronter. Please check the list for your name.
- List will updated next monday and the monday after that.
- List will be sent in to administration in a couple of weeks.
- My office is C129, but I will be away for most of May. Will answer emails.
- Bernt will try to make a zip file of fronter info and give it to reference group. You are allowed to keep a copy for yourself, but don't publish or redistribute.
- If you fail: new exam in August. Fronter closes 1st july.

Below follows a summary of what I think are the most important things we have covered in the course, together with perceived difficulty. This is not a list of what you might get on the exam!

Module 1: Probability

C students:

- Elementary probability.
- Markov chains
  - Concepts
  - Calculations on absorbing chains - using e.g. fundamental matrix.
  - Programming with these concepts (eigen, numpy).

A students:

- Elementary probability in unfamiliar context.
- Markov chains to model something unfamiliar. e.g. a software system.
- Ergodic chains - calculations and analysis.
- Setup a complicated markov chain with code (e.g. like snakes and ladders hand-in - but unfamiliar)

Module 2: Logic

C students:

- Clearing, toggeling, setting bits.
- Counting set bits and similar simple tasks.
- Be able to say something intelligent about backtracking.
- Read some prolog code (up to list manipulation).
- Create and solve knight/knave (and spy) puzzles.
- Read and write "logic with arithmetic"

A students:

- Complicated and unfamiliar bitwise operations.
- Implement backtracking on a problem.
- Write prolog code (up to list manipulation).
- Solve harder puzzles.

Module 3: Matrices

C students:

- Elementary concepts.
- Least squares.
- Eigenvalues/eigenvectors.
- Eigenfaces.

A students:

- SVD.
- Apply knowledge in unfamiliar situation.

Module 4: Interpolation/procedural

C students:

- Lerp, slerp, bezier curves (calculations + implementation).
- Midpoint displacement and diamond square.
- Perlin noise.
- Elementary boids.

A students:

- Boids with field of view calculations. Ideas about optimizations.

Module 5: Path finding

C students:

- Go through the algorithms on example.
- Knowing the difference between A* and Dijkstra.

A students:

- More complicated heuristics.

Module 6: Mechanics

C students:

- Diff/integration on polynomials. Chain rule.
- Gameloops and physics objects.
- Integration methods - implicit/explicit euler.
- Adding forces to a physics simulation (wind, friction etc.)
- Use Newtons equation F = ma.
- 1d collision and 2d collision of spheres (elastic/inelastic)
- Particle systems
- Simple calculations CCD.

A students:

- R2 level integration/differentiation.
- More sophisticated game loops.
- More complicated computations with Newtons equations.
- Implementing collision of spheres (with variations).
- Implement CCD (with variations)
- Some knowledge of different integrators and what the problems are.

Module 6: Functional programming

C students:

- Concepts (e.g. immutable, lazyness, pure functions).
- Solving simple list problems using map, filter, zipWith etc. These functions will be given on the exam.
- Read and write simple lambdas.
- Understand the different notions of functions in C++.
  - plain C functions.
  - function pointers.
  - function objects.
  - lambdas.
  - std::function.

A students:

- Some ideas about how to implement lazyness and immutability in C++
- Solve complicated problems with map, filter, zipWith etc.
- Automatic differentiation
- Understand and can say something intelligent about std::variant, std::optional, std::tuple and std::pair.
- More complicated lambdas, generic lambdas, recursive lambdas.
- General familiarity with problem solving using recursion.

Ideas for next year

- Prolog could be replaced with C++, but requires template programming.
- Matrices will be redesigned and include coding assignments in c++.
- Hand-in with templates, lambdas and functional programming in general. Move functional programming up to earlier date.
- Python - should not be dropped, although it is not working according to plan.
- Possibly 8-10 hours a week - 4 math lectures - 2 programming - 2-4 tutorial (with assistant and/or me). The course is a bit too dense as it is.
- Introduce some more languages (especially if prolog goes out): Suggestions: Haskell, F-sharp, Lisp.

I need 1-2 assitants

Job description:

- Grading (mostly programming) - after I have passed the students.
- Help in tutorials (mostly programming) - you get paid for preparation time.
- Rewrite my old code to make it more readable.
- Maybe some administrative tasks on Blackboard.

Requirements:

- You are a nice and helpful person (so that it is easy for students to talk to you).
- C in this maths course and A in both programming courses, or
- A or B in this maths course.

Rewards: Money (not sure how much) and you will get a reference for your first job applications (i.e. I will accept phone calls from your prospective employers).
I have a list of names already, but you can still send me a non-binding email if you are interested.