SEMESTER II EXAMINATION – 2016/2017 COMP 30540 Game Development Prof. S. Dobson Instructions for Invigilators Written materials and electronic devices are prohibited.

++

0. (Shape collision)

- What are voronoi regions? Illustrate by discussing Voronoi regions surrounding a rectangle in an infinite plane.

```
  - in mathematics, a voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific
subset of the plane.
```

- What is the Minkowski sum of two 2D shapes, and similarly, of two 3D shapes? Illustrate with an example of two off-origin non-identical triangles.

```
  - two sets a⊕b is the minkowski sum of a and b a⊕b = {a+b! a∈a, b∈b}
    - a is a polygon and b is a convex | disc
    - this helps with collition - a is an obsticle object and b is a moving object
```

- Give an algorithm for cheaply computing whether a circle of given radius and centre position overlaps a coplanar axis-aligned RECTANGLE of given size and position.

```
  - what is s and what is d? > s side and d ???
        -     d=0
        -     if   (cx>boxr)    {size=cx-boxr; d+=s*s}      > side
        -     elif (cx<boxl)    {size=cx-boxl; d+=s*s};     > other side
        -     if   (cy<boxb)    {size=cy-boxb; d+=s*s}      > side
        -     elif (cy>boxt)    {siz`e=cy-boxt; d+=*s};     > other side
        -     return            (d - cr*cr)                 > will put it back together
```

- Compose and explain a similar algorithm for cheaply computing whether a given axis-aligned rectangle overlaps a given coplanar square tilted at - Refer to the diagram below:

```
- minkowski sum, difference               => two shapes (area, volumes)
- the calculations of checking all points => doing the additioon with the minkowski sum and subtracting the other shapes
points to find theiur differences
- point-to-face collision vs edge to edge collition
```

- Explain why testing for intersection of two 3D convex shapes may make use of the Minkowski sum of one shape and the reflection-through-the-origin of the other.

```
- the minkowski sum can be used for = collition as an overall into a game by a set of calcultions determinind the outer
edges of a polygon.
- the minkowski sum can be used to  = check the intersections between two 3d convex shapes by doing the subtractions of
the points and the edges of another set of convex.
- once you have the differences in calculations between the sets and have their middle meeting points you can then decifer
what actrions you want the convex to do within the game.
```

++

1. (Less common AI techniques)

- Why might the Sense-Think-Act cycle of a NPC AI be extended to use some learning?

```
  - path-finding
  - FSTNs                              => for scripted behaviours ( perhaps probabilistic transitions for adaptation )
  - VISIBILITY                         => needs to figure out where the objects are
  - HEARING                            => when one character produces noise they need to be able to hear it
  - COMUNICATION                       => speech, gesture, writing
```

- Describe two simple forms of learning that might be incorporated into NPC AI

```
  - path-finding among with :
  - IF THEN                            => rules for finding behavior
  - SEARCH                             =>  route finding and  chaining actions to make plans
  - LEARNING ONLINE AND OFFLINE        => trees, neural networks
```

- Why would AAA development teams choose to avoid more sophisticated learning methods?

```
  -- memory
  - big games also means there coule be way more going on within the game
  - big teams, communication and the complexity of a game. readability
  - keeping it simple helps building more features
```

- Outline an algorithm for processing a 2D map in which passable and impassable pixels are easily distinguished or even labelled, to find choke points where movement might be difficult or impossible.

```
-- voronoi diagram                  =>  gives you three calculations
- vertices of the Voronoi              => region is equo distant too its center point
- corner of the Voronoi                => region is quo distant to its center pooint
- Map data                             => Delaunay Triangulations + Voronoi diagram
- you would need to find the less than the points of the Delaunay Triangulations and check them agains the less then or
greater than of the Voronoi diagram
```

++

1. (Behaviour Trees)

- In a looping animation of a 3D character, some of its poses ought to have particular properties. Explain those properties.

```
- c0 = continuity of position                                                    • but if playing loop more than once, do
not repeat both 1st and last each time
- c1 = continuity of velocity (derivative of position)                           • bone velocities at start and end
should be identical
- c2 = continuity of acceleration (derivative of velocity) – probably not desired    • bone accelerations probably need not
be identical
```

- Give two separate reasons why a character may need to be rendered in a posture that has not been explicitly represented as a keyframe pose.

```
- to figure out its coordinates and predict its next motion and plan of motion
- due to faults in the computers calculations of hard to calculate poses of the character
      - sometimes rendering the character can give new pixels to calculate of the next iteration of where a character might be
>
```

- Explain why with inverse kinematics, the coordinate descent technique may suffer the “2-bone problem”.

```
- when bones are nearly straight there is no angle to figure out what is going to when a connection of two bones
- since with inverse kinematics is to find the logic in which bones are needed to be foung when connected its hard to find a
joint when the leg is straight

- rotations and positions need to be cascaded through tree of bones
```

- What is the difference between forward kinematics and inverse kinematics?

```
- inverse kinematics = the logic befind kenetics to figure out how a bone or a join needs to be working and positioned to get
an end effector
- forward kinematics = the action of how a bone or joint is working > cascading to all bones
```

- What are behaviour trees? What may a leaf node of a behaviour tree consist of?

```
    - describe switchings between a finite set of tasks in a modular fashion.
    - each node of a tree might consist of a behavior that the character attacheed to the tree needs to execute and the child
of that specific node will execute tasks related tot its parent
        - composite                         => these are the nodes that define the root of a branch and the base rules for
how that branch is executed
        - task                              => these are the leaves of the behavior tree, the nodes that "do" things and
don't have an output connection.
        - decorator                         => also known as conditionals. these attach to another node and make decisions
on whether or not a branch in the tree, or even a single node, can be executed.
        - service                           => these attach to composite nodes, and will execute at their defined frequency
as long as their branch is being executed. these are often used to make checks and to update the blackboard. these take the
place of traditional parallel nodes in other behavior tree systems.
        - root                              => the root node is unique in the behavior tree and is the starting point for
the behavior tree. it can only have one connection, and you cannot attach decorators or services to it. the root node has no
properties of its own, but selecting it will show the behavior tree properties in the details panel, where you can set the
blackboard asset of the behavior tree.
```

- Describe the most common types of behaviour tree interior node, in their simplest forms.

```
   - can be used                        => in physics, computer science and game development
   - bspt                               => binary space partion trees - traversal
   - z-buffering                        => x & y (or h and v). add a z : for each pixel on screen, trace a line into the
simulated world then check the distance between the both
   - trees                              => decision trees vs behaviour trees : data driven vs code driven
   - decorator vs other leaf
```

- Why are less simple forms of those interior node types sometimes used?

```
    - in case of drastic changes within a game, for scaling purposes a player would neeed to become more complex.
```

- So-called Decorator nodes are also sometimes. Give examples of their role in a behaviour tree.

```
      - leaf is a set of actions decorators are a new set (parent) that holds other lefs and sets of behavior
```

- Why may errors arise in using behaviour trees, and how are they accommodated?

```
    - 1 memory and 2 conflicts in its environment
    - errors in behavior trees maybe produced by a mixture of memory and graphics glitch which stops the tree in its
proceedural tracks.
    - which then could freeze a character, its behavior and the artifacts in the map.
    - a solution to this problem would be to be able to proof check the tree for errors that arise in its environment
```

++

2. (Physics)

    - How does a physics-based approach help in modelling the behaviours of non-rigid 3D bodies and of colliding rigid bodies?

        - since we are always out to fing a realistic aproach to what we design
            - it is important to gather the right calculations to the movement of what we think the objects momentum is. >
        - this means accuratly calculating attributes like an objects force,
            - its acceleration, its drive, gravity and other physics based movents. >
        - physics also helps in other ways such as making an object stand out with
            - pong surface vs zbuffering to slowly gather the data of what you are rendering >
        - another point we coud say is that within a 3d grid you need to calculate an objects
            - x, y and z entourage. helping the process by modeling an oo that has the potential of physics effects >
        - non-rigid is something that will change its shape!!!! a rigig mody is a model that has the ability to transform - like liquid
            - force on a non rigid body will not change the shape of the body where a non rigid body will change depoending on the force its applied to as an effect >
            - this may be affected by - cloth, fluids, brittle objects, >

    - Discuss how modelling two distinct types of non-rigid 3D bodies may be done using particles and links,

        - describing any properties that should be associated with particles and links.

```
- links between vertices in physics mesh can be associated with natural length, tensile strength, compressibility,
stiffnes
- depending on an objects force, the links are applied a sertain type of springiness that is calculated to produce a
collition of soft body deformation and movement between the two rigid bodies.
- particles with mass, centre of mass, moments of inertia : relate torque to angular acceleration; one per axis

- links will be depending on thge outerbounds in which connect one object to another in a rigid body vs a particle
will be the attribute to the force its given
- newton's laws of motion, numerical integration is required
```

    - Why does physics simulation rest on numerical integration? What dangers attend the use of numerical integration?

        - force f yields acceleration a => (angular accelerations too)
        - integrate a to get velocity v => (rotational velocities too)
        - integrate v to get position p => (rotational positions too)
        - the over exageratioon of these calculatioons can cause
            - unrealistic collision behaviour => unrealistic : collision
            - inaccurate friction behaviour => unrealistic : friction
            - unrealistic oscillations => unrealistic : oscillations
            - framerate drop => unrealistic : graphics issues
            - game crash => unrealistic : overload

++

3. (BSP) - fairly expensive operation, O(N.logN)

    - What is a Binary Space Partitioning (BSP) of a set of polygons representing surfaces in a 3D scene?

```
- is a method for recursively subdividing a space into two convex sets
- in a 3d scene bsps are used to calculate the : painter's algorithm
      - furthest polygons first and work backwards - each frame >
      - issues with frames objects moving >

- z-buffering
      - for each pixel on screen, trace a line into the simulated world >
      - poly is found, compare its distance with the closest known >
      - if closer, update closest distance and paint that pixel >

- the basic idea is to partition all polys in a scene into particular pivot polys
  - with a bring to front and to rear
  - polys that intersect a pixot are then cut into two
  - polys to front are recursively partitioned using one as pivot; likewise, polys to rear.
```

- explain how a bsp allows realistic rendering of such polygons without sorting polygons by distance from a camera and without z-buffering.

```
  - what happends after they are devided in two? ---- is what the question is asking
  - viewpoints, partial occlusions, perspective, haziness, reflections, stereo vision
  - since bsp trees main purpose is to subdevide trees into sets the operation can be done as many times as needed
     - this means that since each convex can be taken as data after calculating their possition, the convexes are easily
manipulated making it easy to gather its finest pixel >
```

- give two reasons why forming a bsp is facilitated if only triangles are used, and not general polygons.
  - triangles are easily devided in two so a bsp tree can gather its counter edges as two seperate nodes over time
  - if a plane-parallel triangle is coplanar, it must be added to a pivot set
- it is desirable that each distinct object should have its surface polygons represented in its own bsp tree. why is this so? when, and how often, should a bsp for an entire scene be created?

```
  - reduce confustion between objects and bsp trees when manipulating a big game
  - reduce since a specific object is normally connected with its own vertices
        - and the oragism of a bsp tree is based on its closest nodes on a surface it is economical to gather its
surounding object and encapsulate it >
        - this being said bsp trees can eventually be programmed to mesh with other bsp trees to gather more sets of
vertices >
```

- why is it useful to associate bounding boxes with bsp tree nodes, in addition to the polygons they contain?

```
  - obbs give the bstp tree a way to detect specific outer edges of objects
  - obbs are easily computed from their pivot poly(s) and the obbs of their to - front and to-rear bspts
      - (although they quickly grow very large if done this way) >

  - if you axis-align the bounding box with the root node w
  - distance to origin -  t (with vertices a, b, c)
```

++

4. (Animation)
   - what are the processes of rigging and skinning in 3d character animation?
     - creating the mesh, assembling bone structure, assigning specific nodes to joints for manipulation
   - how is the orientation of an entire 3d object represented using : EULER ANGLES? > x, y, z
     - x and y are the cordinates that are based on a 2d spectrum with a center point. at this center point is where we have another stem called z
     - z main objective is to break out of the x y patters and give depth to an object where a user can see a mesh in a rotating axis
   - how is the orientation of an entire 3d object represented using : AN EXPONENTIAL MAP?

```
  - an exponential map gives a flat surface layout of the entirety of an object. therfore easily able to pin point where a
section is within an object.
  - conversion of 3d space into 2d. only need to work with x and y to find its desired location and cascade it back to its
original 3d form
  - not always accurate - it exaggerates the sizes of areas
```

   - how is the orientation of an entire 3d object represented using : A NORMALISED QUATERNION? - x, y, z, w • w gives cos(θ/2)
     - a normalized quaternion moves along a unit sphere - its first time derivative is tangent to that sphere.
     - quaternions: represent a rotation as a 4-tuple -x, y, z, w> - w stands for • w gives cos(θ/2)
     - cos = in a right angled triangle, the cosine of an angle is: t - he length of the adjacent side divided by the length of the hypotenuse. - top and bottom of a triangle (/)
     - ways to interpolate normalised quaternions

```
- lerp ≈ nlerp: weighted average of each component individually, normalise result
      - lerp is a linear interpolation so that the distant between each step is equal across the entire
interpolation.>
- slerp (spherical interpolation): calculate great circle between start & end
      - slerp is a spherical linear interpolation.>
- log-quaternion lerp (aka exponential map interpolation) : exotic & complex, described in 1998 paper “practical
parameterization of rotations using the exponential map”
```

   - what are the ideal properties of an angular interpolation algorithm?
   - in what ways do nlerp and slerp applied to quaternions fall short of having these ideal properties?
     - interpolation in games takes place between relatively close orientations - difference is less than 45° nlerp speed is almost indistinguishable
       - so nlerp, being simplest, is therefore preferred
     - interpolation should be along the shortest path (“least torque”, least twisting)
       - exponential map interpolation doesn’t achieve this

- interpolation should proceed at constant speed
  - nlerp doesn't achieve this
- order of blending three rotations should not matter
  - slerp doesn't achieve this

- why is nlerp accepted in practice for angular interpolation of quaternions?

++

5. (Audio)

- digitally sampled audio is used for much of the sound played during games. explain how features of human hearing, and the nyquist limit, are used to choose sampling rates for digital sampling.

```
  - the nyquist frequency is the maximum frequency that can be sampled without aliasing
  - sample rates are bits you would like your samples to take
```

- explain simple processing that can be performed on sampled mono sound to give the illusion of sources being situated in, and/or moving in, a 2d or 3d game level.

```
  - stereo vs mono and how mono is a one channel and can be changed with a pan gnob ...
```

- discuss briefly how timing and head modelling can be used to provide more convincing 3d audio effects using sampled sound. - HRTF (HEAD RELATIVE TRANSFER FUNCTION)

```
  - with hrtf (head relative transfer function)
  - two speakers/earphones can mimic 3d sound origin quite successfully
        -  especially with earphones
  - but it can be frustrated for several reasons
        -  variety of consumers' speaker arrangements (eg surround sound) -  room in which sound is played -
individual's ears
```

- discuss how aspects of the simulated world in which sounds travel from source to ear might need to be modelled for a convincing audio experience.

```
  - like a 3d theater can have its surroundings and be good with its environnent
       - our ears perceive the same thing - being a 7.1 sound surround. talk about base and how its the same in the game
>
  - rumble, high freequencies for perceived animals and lower freequencies
       - to decifer between human voices and ranges >
```

++

6. (Binary Space Partitioning)

- explain how a binary space partitioning tree (bspt) organizes a set of polygons representing the exterior surface of a solid 3d game object.
  - fairly expensive operation, o(n.logn)
  - pivot poly and any others coplanar with it = polys to front are recursively partitioned using one as pivot; likewise, polys to rear.
    - polys that intersect pivot's plane are cut into two, one behind, one in front
    - if not convex, or if you insist on triangles, then maybe more than two

- how is a bspt used to decide the order of drawing polygons onto a display?

```
  - if "from front"   = (recursively) paint the scene "to rear" of it
  - if "from rear"    = paint the "to front" partition
  - if edge-on        = paint "to front" and "to rear" partitions in either order & ignore it and those coplanar
```

- explain how the bspt for an entire scene can be built by merging bspts each representing individual objects and/or the environment in which they are positioned.

```
  - obb , btspt - the combination of merging - binary space partitioning

  - painter's algorithm -  paint furthest polygons first, overpaint with progressively nearer ones
  - binary space partitioning tree !!!!!
  - z buffering - for each pixel on screen, trace a line into the simulated world
    - if closer, update closest distance and paint that pixel
```

- how does the bspt organization of polygons assist in determining whether the movement of one object brings it into collision with another object?

```
   - by checking the xyz coordinates of the 3d object from the pivot point of where an actual node you can get the
per/poly collision info through the subdevision
   - you can also get the collition information based on where the xyz cordinates meet the calculations of another bsp
trees organization to distinct collition
   - trees should not necessarily be balanced for best result on collition
```

- why is it preferable to represent the volume of all the polygons represented within a bspt using a convex hull rather than using an object-bounding-box aligned with the normal to its pivot plane?

```
   - object-bounding-box
      -  can calculate very well its outer edges but are constrainted to it only. this might be better for 2d games
    - convex hull
      -  much less easy to compute, but most economical in spatial terms
      -  with well-chosen pivot planes (such as those on objects' convex hulls) good approximations to their overall shape
can be got by limiting the depth of tree that is considered.
      -  its important to realize how a leaf is bounded by its parent nodes to create an atomic space.(convex hull in
other words).

      -  notes: voxels – divide 3d space into uniform cubes, like pixels divide 2d into squares
```