SD1-A8: Adventure Prototype

Continue the development of your SD Engine and Adventure game technical prototype.

# Requirements

Be sure to consult the usage case examples provided in Canvas for implementation details and assumptions regarding these features.

Note that this assignment offers a total of 120 points, to be taken out of 100, with a maximum grade of 100/100. Thus, many of the requirements are meant to be optional, since you can skip up to 20 points’ worth and still get a perfect score.

1. (10 points) Add mouse cursor support to InputSystem
2. (5 points) Convert RandomNumberGenerator to use RawNoise.cpp,hpp instead of rand()
3. (5 points) Add ActorDefinition class, constructs from XML; Actors construct from an ActorDefinition\*
4. (5 points) Create multiple Actors of different types, wandering around, one of which is player-controlled
5. (5 points) Actors draw using a .75 aspect sprite; local bounds, anims, and walk/fly/swim defined in XML
6. (5 points) Add “debug inspection” with mouse for Actors and Tiles (show info for actor or tile hovered)
7. (20 points) Add a simple EventSystem; register (subscribe) functions and fire (publish) by string-name
8. (40 points) Add a data-driven step-based procedural map generation pipeline, with 4+ MapGenStep types; black types are “required” (worth more points), blue types are “bonus” (worth fewer points, or none):
   1. (5) MapGenStep\_**Mutate**: modify all qualifying tiles
      1. *No additional custom attributes; supports all base class attributes (see requirement 9)*
   2. (7) MapGenStep\_**Worms**: modify tiles visited in N random “drunken” walks of M steps each
      1. numWorms=”5~10” // how many “worms” are spawned
      2. wormLength=”5~10” // how many drunken steps each worm takes before
      3. startType=”” // each worm starts on a randomly-selected tile of this type
      4. startTagsInclude=”” // worms start only on tiles with these Tags
      5. startHeatWithin=”-9999999,9999999” // worms start only on tiles with this Heat
   3. (9) MapGenStep\_**FromImage**: modify tiles using a named .PNG image with tile-type texel colors
      1. imageFilePath=”” // required! (error if not specified) – image to use as map data
      2. alignmentX=”0.0~1.0” // horizontal alignment of image tiles within map; 0.5 = centered
      3. alignmentY=”0.0~1.0” // vertical alignment of image tiles within map; 0.0 = at bottom
      4. rotations=”0~3” // number of +90 degree rotations to apply to image
      5. mirrored=”0~1” // whether or not to also mirror (x-reverse) image tiles
   4. (9) MapGenStep\_**CellularAutomata**: modify tiles based on their neighboring tiles
      1. ifNeighborType=”” // neighboring tiles must have this type to qualify
      2. numNeighbors=”1~9999999” // # of qualifying neighbors required to modify this tile
      3. ifNeighborTagsInclude=”” // neighboring tiles must have these tags to qualify
      4. ifNeighborHeatWithin=”-9999999,9999999” // neighbor tiles need this heat to qualify
      5. ifNorthNeighborType=”” // north (east, etc.) neighbor must be of the named type, etc.
      6. ifNorthNeighborTagsInclude=”” // north (east, etc.) neighbor must have these tags
      7. ifNorthNeighborHeatWithin=”-9999999,9999999” // north neighbor must have this heat
      8. etc. for East, West, South, possibly also Northeast, Northwest, Southeast, Southwest
   5. (3) MapGenStep\_**PerlinNoise**: modify tiles based on Perlin noise values
      1. ifNoiseWithin=”0~1” // Perlin noise in [-1,1] must be within this range to qualify
      2. gridSize=”10~30” // Perlin noise base-octave grid size (noise period/wavelength)
      3. numOctaves=”2~4” // # of octaves (double-frequency, half-amplitude) to sum
      4. persistence=”0.5” // amplitude scale for each successive octave
      5. octaveScale=”2.0” // frequency scale (grid size divisor) for each successive octave
      6. seed=”0” // seed offset (added to maps’s base RNG seed) for Perlin noise seed
   6. (2) MapGenStep\_**RoomsAndPaths**: modify rectangular areas of tiles with paths connecting them
      1. <Rooms> element attributes:
         1. count=”3~7” // number of “rooms” to spawn
         2. width=”4~8” // width (range, in tiles) for each room
         3. height=”4~8” // height (range, in tiles) for each room
         4. numOverlaps=”0~0” // number of times rooms are allowed to overlap
         5. floorType=”” // tile type to set for room interior (“floor”) tiles
         6. wallType=”” // tile type to set for room exterior (“wall”) tiles
      2. <Paths> element attributes:
         1. pathType=”” // tile type to set for paths connecting rooms
         2. loop=”false” // whether to connect the last room back to the first room
         3. straightness=”0~1” // how straight (e.g. axis-aligned) each path should be
         4. numExtraPaths=”0~0” // how many extra room-to-room paths to generate
   7. (2) MapGenStep\_**SubMap**: run an entire named MapDefinition sequence in a sub-rect of this map
      1. mapDef=”” // required! (error if not specified) –
      2. subBoundsWidth=”0.5~1.0” // fraction of map width to run sub-definition within
      3. subBoundsHeight=”0.5~1.0” // fraction of map height to run sub-definition within
      4. subBoundsAlignmentX=”0.0~1.0” // u-alignment of sub-definition area within map
      5. subBoundsAlignmentY=”0.0~1.0” // v-alignment of sub-definition area within map
   8. (1) MapGenStep\_**BoundingBox**: modify tiles whose centers fall within a defined OBB2 area
   9. (1) MapGenStep\_**DistanceTaxicab**: modify tiles with taxicab distance(s) from a given point
   10. (1) MapGenStep\_**DistanceCircular**: modify tiles with centers within float-range dist from a point
   11. (?) MapGenStep\_**SomeCoolIdeaThatYouCameUpWithAndApprovedWithTheProfessor**: ???
9. (25 points) Add the following XML attributes on the MapGenStep base class (i.e. usable by all MapGenStep subclass types); black attributes are “required” (worth more points), blue attributes are “bonus” (worth fewer points):
   1. (4) **iterations**=”1” // number of times to execute this step sequentially (each one independently)
   2. (4) **chanceToRun**=”1.00” // probability of this entire generation step even running (vs. skipping)
   3. (4) **chancePerTile**=”1.00” // probability for each tile to be considered by this step
   4. (4) **ifType**=”” // only modify tiles which are of the named type (any type okay if unspecified)
   5. (4) **setType**=”” // all tiles affected by this step have their type changed to this type (if specified)
   6. (2) **setTags**=”” // all tiles affected by this step have these Tag(s) set (and/or cleared) on them
   7. (2) **ifTagsContain**=”” // only modify tiles which contain all of these Tags (comma-separated list)
   8. (1) **setHeat**=”” // if used, each tile affected gets a random “heat” value from this float range
   9. (1) **ifHeatWithin**=”-9999999~9999999” // only modify tiles whose “heat” is within this range
   10. (1) **addHeat**=”0~0” // each tile affected by this step adds a random (float range) amount of heat
   11. (1) **subtractHeat**=”0~0” // each tile affected subtracts a random (float range) amount of heat
   12. (1) **multiplyHeat**=”1~1” // each tile affected multiplies its heat by a random (float range) value

# Submission

Submit your assignment by following the instructions above and checking in all the required files to Perforce (including a Release-built Adventure\_x64.exe), with the check-in comment “SD1-A8: COMPLETE” for the changelist you want me to grade. My Perforce changelist # **179863**

Make sure you **Add** and **Submit** any **data assets** your game requires (in the Run/Data/... folder tree).

Also, in Canvas, you should submit a .zip file as follows:

* Submit a single .zip file to Canvas under the assignment.
* Your .zip should be named: **C28\_SD1\_A8\_LastnameFirstname.zip**

*For example, Jane Smith would submit a file named* ***C28\_SD1\_A8\_SmithJane.zip***

* Your assignment submission .zip file should contain the following:
  1. A **video recording** of you playing your game (and showing your code, as needed); be sure to visually demonstrate and verbally narrate each feature you want credit for
     + The video should be: 1920x1080 **.mp4** at 60 FPS, under 5 minutes and < 100 MB
     + Recommend you use OBS Studio (64bit) to record; make sure you check (watch) the video!
     + See the accompanying Demo video for an example of what’s expected here
  2. A **copy of this Word document**, with the following modifications:
     + Your submitted Perforce changelist # entered at the top of this section, at “My Perforce changelist # **???**” – this is the Perforce changelist # I should Get, run, test, and grade
     + Each line in the “Requirements” section with (X points) **must** be highlighted:
       - Fully completed requirements are highlighted cyan
         * for features you believe you’ve met/reproduced nearly exactly
       - Partially completed requirements are highlighted yellow
         * for features done but lacking or differing significantly vs. demo
       - Missing requirements are highlighted red
         * for features not implemented (not working or not attempted)