Civ1 Map Generation explained

Discussion in 'Civ1 - General Discussions' started by darkpanda, Jun 19, 2013.

Page 1 of 2 1 2 Next >

Watch Thread

darkpanda
Dark Prince

Joined: Oct 28, 2007
Messages: 600

This thread is meant to document the internal routines used by CivDOS to generate new, random maps.

To generate new maps, CivDOS uses a set of **4 input parameters**, whose value ranges from 0 to 2:

- **Land mass**: this controls the total land mass of the map, i.e. the total number of non-ocean squares (arctic/antarctic excluded)
  - 0: Small - islandish world
  - 1: Normal - balanced islands and continents
  - 2: Large - pangeaish world
- **Temperature**: this controls the span ratio of arid areas (deserts) around the equator
  - 0: Cool - more tundra/artic
  - 1: Temperate - balanced
  - 2: Warm - more deserts
- **climate**: this controls the overall ratio of "wet lands" (jungle, swamp)
  - 0: Arid - more deserts
  - 1: Normal - balanced
  - 2: Wet - more jungle/swamp
- **Age**: this controls the erosion of the landscape (hills, mountains, inland seas & lakes)
  - 0: 3 billion years - more mountains
  - 1: 4 billion years - balanced
  - 2: 5 billion years - more hills/lakes

When selecting "Start a New Game", **CIV sets the parameters to [1,1,1,1] by default**.

The map generation process goes through the following phases:

- **Land mass generation**: the shape of continents is generated randomly, depending on the custom-selected land mass, with additional processing to refine the continent shapes
- **Temperature adjustment**: the terrain types are assigned based on land mass and temperature (desert, plains, arctic, tundra, hills, mountains, ocean)

- **Climate adjustement**: the terrain types are re-assigned based on climate (jungle, swamp, forests, arctic, plains)
- **Age adjustment**: some more rough touches to terrain types, based on age/erosion (lakes, inland seas, mountains, hills, ...)
- **River generation**: creation of rivers...
- **Computation of map-related data**: continents/seas segmentation, sizes, building sites, high-level pathfind...
- **North and South poles generation**: creation of poles

A. Land mass generation

The land mass generation uses **2 different map areas**, each containing 80x50 squares:

- the **geography area** is where the map is progressively built; this is the "layer 0" of a MAP gamesave (see this post)
- the **stencil area** is where *chunks of land mass* are randomly spawned before being merged into the geography area; for this, CIV uses what is normally the "layer 8" of a MAP gamesave (see this post)

The land mass generation routine is as follows:

1. Clear the stencil area, then **generate a random chunk** of land mass in the stencil area
2. **Merge the random chunk** to the geography area
3. If **land mass is not sufficient**, loop back to step 1.
4. Fix **narrow passages**

Each of the above steps is detailed hereunder:

1. Generate a random chunk

The random chunk algorithm is:

1. Initialize:
   - Initialize the whole stencil area to **value 0** (empty / no land)
   - Select a **random starting position** (x,y) in the range (4,8)-(71,33)
   - Select a random **path length L** in the range [1-64]
2. Set the 3 positions **(x,y), (x+1,y) and (x,y+1)** to **value 15** (land); this is like painting position (x,y) with a Gamma-shaped brush
3. Randomly choose the **next position to paint within the 4 direct neighbours** of (x,y) at (0,-1)-North, (1,0)-East, (0,1)-South and (-1,0)-West
4. Decrement path length L by 1
5. Loop back to step 2. above until either path length L = 0 or (x,y) falls outside the bounded range [3,3]-[76,45]

2. Merge the chunk to main geography

To merge the chunk into the main geography, the algorithm is that for each square (x,y) from the chunk:

- If the corresponding geography is empty (value 0), **set its value to 1**
- If the corresponding geography is *not* empty, **increment its value by 1**

This step can be viewed as if the chunk area was a small piece of earth crust *pushing up from the ocean floor*: whatever land is already there is pushed up 1 level as well.

3. Check whether land mass is sufficient

There is simple formula to verify whether the map contains enough squares compared to the **land mass parameter**:

Spawn/merge new random chunks **while (totalLandMass < ((landMassParam + 2) * 320)**

This land mass *threshold* has the following values for different values of the land mass parameter:

Column 1	Column 2
land mass param	minimum squares
0	640 squares
1	960 squares
2	1280 squares

4. Correction of narrow passages

What I call a *narrow passage* is this famous 2x2 square configuration where **ocean and land squares form an X**, that can be seen in the EARTH map:

Code:
```
####~~~~   ~~~~####
LAND~~~~   SEA~####
####~SEA   ~~~~LAND
####~~~~   ~~~~####
~~~~####   ####~~~~
SEA~####   LAND~~~~
~~~~LAND   ####~SEA
~~~~####   ####~~~~
```

This land configuration is never present in random CIV games precisely because CIV removes them at this step, by replacing them as follows:

Code:

```
####~~~~          ########
LAND~~~~          LAND####
####~SEA          ####LAND
####~~~~   ---\   ########
~~~~####   ---/   ########
SEA~####          LAND####
~~~~LAND          ####LAND
~~~~####          ########

~~~~####          ~~~~####
SEA~####          SEA~####
~~~~LAND          ~~~~LAND
~~~~####   ---\   ~~~~####
####~~~~   ---/   ########
LAND~~~~          LAND####
####~SEA          ####LAND
####~~~~          ########
```

To illustrate the whole Land mass generation process, hereunder are some animated GIFs:

B. Temperature adjustments

The temperature adjustment phase does the following:

- Define **initial elevation terrain** based on values coming from the land mass generation: ocean (value=0), flat land (value=1), mountains (value=2) and hills (value>=3)
- Set **initial land types to desrt, plains, tundra or arctic** depending on the Temperature parameter, using the following logic, **for each land map square (x,y)**:
  - **Compute the map square latitude with some randomness** (vertical distance from the equator), and taking into account the temperature parameter, as follows:
    - Substract 29 from y
    - Add a random value in the range [0..7] to y
    - Unsign y value
    - Add (1 - temperatureParameter) to y
  - **Rescale to [0..7]** the value above, using the formula latitude = latitude / 6 + 1
  - **Assign the land type** as follows:
    - if latitude = **0 or 1 -> DESERT**
    - if latitude = **2 or 3 -> PLAINS**
    - if latitude = **4 or 5 -> TUNDRA**
    - if latitude = **6 or 7 -> ARCTIC**

To illustrate the **temperature adjustment process**, I designed a "blocky" land mass with some elevation, and applied the temperature adjustments for values 0, 1 and 2 of the temperature parameter. The results are shown below:

Column 1	Column 2	Column 3
land mass		
temperature 0	temperature 1	temperature 2

C. Climate adjustments

The climate adjustment process war a real pleasure to decipher and understand: somehow it is mimicking the real-world water cycle, accumulating water from the ocean into clouds, which then downpour into rain and shrink as they fly across over earthland...

The exact routine is as follows, processing each map ROW (latitude) twice, from West to East, then from East to West:

- For each map row (y from 0 to 49):
    - Initialize the **row "wetness" to 0** (wetness can be seen as clouds accumulating humidity...)
    - Initialize the **row "latitude" to |25-y|**, i.e. distance from the equator
    - Firstly, process each row square **from West to East** (x from 0 to 79):
        - If the square is **OCEAN**, accumulate clouds:
            - **compute the square "wetness yield"** as **|latitude-12|+climateParameter*4 -** *note: somehow, this yield is the highest for latitude 12, i.e. temperate regions between poles and equator; it is also boosted by the climate parameter, quite logically*
              *[*] if the square wetness yield is **higher than the current row wetness**, then **increase the row wetness by 1***

          *[*] Else (if the square is **not OCEAN**), consume clouds:*
            - ***compute a random "rainfall" value between 0 and (7 - climateParameter*2)**; note: we see here that for wet climates (param = 2), the high random bound is lower (7-2*2 = 3), so the consumption of wetness to transform a square is lower as well, and more squares can be humidified...*
            - ***substract the rainfall value from the row wetness***
            - ***convert the land type** as follows:*
                - *PLAINS becomes GRASSLAND*
                - *TUNDRA becomes ARCTIC*
                - *HILLS becomes FOREST*
                - *DESERT become PLAINS*
                - *MOUNTAINS do not change, but additionally **decrease the wetness by 3***

*[*] Reset the **row "wetness" to 0***
*[*] Then, reprocess each row square **from East to West** (x from 79 to 0):*

- *If the square is **OCEAN**, accumulate clouds:*
    - ***compute the square "wetness yield"** as **latitude/2+climateParameter** - note: this yield is different from the previous one, somehow it the highest at the poles...*
    - *if the square wetness yield is **higher than the current row wetness**, then **increase the row wetness by 1***
- *Else (if the square is **not OCEAN**), consume clouds:*
    - ***compute a random "rainfall" value between 0 and (7 - climateParameter*2)**; note: same as previously*
    - ***substract the rainfall value from the row wetness***
    - ***convert the land type** as follows:*
        - *SWAMP becomes FOREST - note: this cannot really happen, since the first pass does not create any swamp...*
          [*] PLAINS becomes GRASSLAND
          [*] GRASSLAND becomes JUNGLE (for latitude<10) or SWAMP (for latitude>=10), and comsume an additional wetness of 2 (row wetness decreased by 2)
          [*] HILLS becomes FOREST
          [*] MOUNTAINS become FOREST, and consume an additional wetness of 3 (row wetness decreased by 3)
          [*] DESERT become PLAINS

*In order to **illustrate the climate process**, I built a special landmass with ovals, and tested the effect of the climate parameter, as shown below:*

Column 1	***Column 2***	***Column 3***
land mass		
climate 0	*climate 1*	*climate 2*

D. Age/erosion adjustments

Next step, the ***erosion process*** *will give the map touches of "real-world feeling" by basically roughing up everything that has been generated so far.*

The algorithm is pretty simple, and does not really seem to follow any logic, more like "randomize" map squares a little bit:

- ***Compute the erosion loop count*** *with the formula:* ***loopcount = 800 * (1 + ageParameter)****; note: this makes between 800 and 2400 loops of erosion, the older the world, the more erosion there is*
- *For loop from 0 to loopcount, do:*
    - *If* ***loop number is even*** *(rightmost, least significant bit is 0), select a* ***random map square*** *to process*
    - *Else, if* ***loop number is odd*** *(rightmost, LSB is 1), select a* ***random neighbour from the previous square****'s inner circle (8 direct neighbours) to process*
    - ***Modify the selected square*** *as follows:*
        - *FOREST becomes JUNGLE*
        - *SWAMP becomes GRASSLAND*
        - *PLAINS becomes HILLS*
        - *TUNDRA becomes HILLS*
        - *RIVER becomes FOREST (cannot happen as no river has been created so far)*
        - *GRASSLAND becomes FOREST*
        - *JUNGLE becomes SWAMP*
        - *HILLS becomes MOUNTAINS*
        - *MOUNTAINS becomes OCEAN, only if NW, NE, SW and SE squares are NOT ocean*
        - *DESERT becomes PLAINS*
        - *ARCTIC becomes MOUNTAINS*

Hereunder are some results illustrating the effect of erosion, for age parameters 0, 1 and 2:

Column 1	***Column 2***	***Column 3***
land mass	*before erosion*	
age 0	*age 1*	*age 2*

E. River generation

Understanding the river process was a little more challenging, and there's part of it that is still a *mystery to me - but it is not directly related to the generation of the map geography... Rather, it is the code that writes pixels with value '8' inside the 'improvement' layer. The purpose of this value has yet to be understood.*

Anyway, for the ***generation of rivers*** *themselves, here is how it goes:*

- *Initialize the* ***river count to 0****; this is the number of rivers generated*
- *The routine below either* ***loops for a maximum of 256 times*** *or* ***stops as soon as the number of generated rivers is above ((climateParam + landMassParam)*2 + 6)****:*
    1. *First* ***create a backup copy of the whole map****, which can be restored later if the river generation fails*
    2. *Initialize the* ***river length to 0***
    3. *Initialize* ***variable A to a random value in the range [0..3]*** *then* ***multiply it by 2****; this makes the value of A taken randomly within {0,2,4,6}; thanks to* ***Renergy*** *for helping fix this*
    4. ***Randomly select a HILLS*** *square on the map*
    5. *Set the currently selected map square to RIVER type*
    6. *Check whether any of the* ***4 direct neighbour squares is of type OCEAN****, that is N, E, S or W, and store this info as a flag 'ocean nearby'*
    7. *Initialize* ***variable B to the same value as A***
    8. *Compute* ***random variable C in the range [0..1]***
    9. *Update* ***variable A as: A = (((C - riverLength%2)*2 + A) & 0x7)****; as* ***Renergy*** *thoughfully put it, this is actually a "river downflow" heuristic for choosing the next river square, that can be expressed as: randomly choose to either go ahead or turn at a right-angle; if river length is even, the right-angle turn is to the right (clockwise), otherwise if the river length is odd, the right-angle turn is to the left (count-clockwise)*
    10. *The next step is the mystery one:*
        - *Set* ***B = 7-B***
        - *If* ***(B <= A) then set the Improvement layer square (x,y) to value '8'*** *; I can't exactly figure out the meaning of this yet...*
    11. ***Increment the riverLength by 1***
    12. *Select* ***neighbour map square with id (A+1)****; as A is in the range [0..7], A+1 is in [1..8] and represents 1 of the 8 direct neighbours of the current square*
    13. *If the* ***previous map square was NOT next to an OCEAN, and the currently selected map square is neither OCEAN, RIVER or MOUNTAINS, then loop back to step 5.***
    14. *Else if the* ***(previous map square WAS next to an OCEAN, OR the selected map square is RIVER) AND the riverLength is equal to or above 5****:*
        - ***Increase the River Count by 1***
        - *Transform all* ***FOREST squares into JUNGLE, within the 7x7-square*** *area centered on the map square*
    15. *Else, discard the current river, and* ***rollback the map geography*** *to its previous state*

This process is a bit hard to apprehend at first, but gets clearer after several passes... I also belive there may be additional processing afterwards regarding the '8' values written in the Improvement layer...

Hereunder is a series of pictures illustrating the 5-step process from Land Mass generation to

River:

Column 1	**Column 2**	**Column 3**
Land mass	*Temperature*	*Climate*
Erosion	*Rivers*	

F. Computation of map-related data

Before going to generate the North and South poles squares, Civ computes the following the map-related data, most of which is eventually stored in the SVE gamesave:

- ***Continents and oceans** sizes, in number of map squares: see the SVE data reference*
- ***Continents path-finding data**: see this post*
- ***Oceans path-finding data**: see this post*
- ***Land values** and **per-continent build site count**: see this thread and the SVE data reference too*

G. North and South poles generation

The creation of North and South poles, otherwise mentioned in this thread from Gowron that provides a tool to stop generation of poles, uses the following, very basic algorithm:

- *Set the **full rows of map squares at latitude 0 (North) and 49 (South) to ARCTIC (value 15)***
- *Repeat the **below routine 20 times**:*
    - *Set a **random square in range [0..79] at latitude 0 to TUNDRA (value 7)***
    - *Set a **random square in range [0..79] at latitude 1 to TUNDRA (value 7)***
    - *Set a **random square in range [0..79] at latitude 48 to TUNDRA (value 7)***
    - *Set a **random square in range [0..79] at latitude 49 to TUNDRA (value 7)***

One more set of pics illustrating the full process:

Column 1	**Column 2**	**Column 3**
Land mass	*Temperature*	*Climate*

Column 1	**Column 2**	**Column 3**
Erosion	*Rivers*	*Poles*

Some pictures are erroneous and contain 'diagonal' rivers; those were created before fixing the random river error identified by Renergy further below

JCivED - a toolbox for Sid Meier's Civilization (MS-DOS)

darkpanda, Jun 19, 2013 Report #1 Like + Quote Reply

---

darkpanda
Dark Prince

Joined: Oct 28, 2007
Messages: 600

Just added details about **Temperature adjustments** above, the remaining details are is on their way 🙂

JCivED - a toolbox for Sid Meier's Civilization (MS-DOS)

darkpanda, Jun 20, 2013 Report #2 Like + Quote Reply

---

darkpanda
Dark Prince

Joined: Oct 28, 2007
Messages: 600

One more word to notify that **climate** and **age/erosion** processing have been added too... And then I realized **rivers** are missing 🙂

Seems like I'm not done with this yet...

JCivED - a toolbox for Sid Meier's Civilization (MS-DOS)

darkpanda, Jun 20, 2013 Report #3 Like + Quote Reply

---

darkpanda
Dark Prince

Joined: Oct 28, 2007

Okay, **Rivers** are now in, it was faster than expected... Next should be North and South poles, but I may take more time to update this, as I don't see much value in it, really.

JCivED - a toolbox for Sid Meier's Civilization (MS-DOS)

darkpanda, Jun 21, 2013 Report #4 Like + Quote Reply