09/11/20 04:42:00 C:\Users\Nathan Longhurst\Documents\Windows Root\Houdini (old)\BIIG Framework\Git Repository (contains copies, and modifications from 28 08 2020 11pm)\lod_and_bake.py

```
import os
import hou

from big_framework import string_processor
from megascans_fixer import os_path_join_fix


def check_node_exists(a_node_path): # I was thinking of doing something like this. Currently I haven't used this - still deciding
    a_node = hou.node(a_node_path)
    if a_node == None:
        raise Exception("A node does not exist at the path: {}".format(a_node_path))

def check_path(a_path):
    if os.path.exists(a_path) == False:
        raise Exception("A file does not exist at the path: {}".format(a_path))

class Bake:
    # makes sense that these are class variables, should these be treated as constants (and hence capitals)?
    map_name_and_houdini_parameter_name_dict = {"Tangent-Space Normal" : "vm_quickplane_Nt", "Displacement" : "vm_quickplane_Ds", "Vector Displacement" : "vm_quickplane_Vd", "Tangent-Space Vector Displacement" : "vm_quickplane_Vdt", "Occlusion" : "vm_quickplane_Oc", "Cavity" : "vm_quickplane_Cv", "Thickness" : "vm_quickplane_Th", "Curvature" : "vm_quickplane_Cu"}

    maps_to_bake_dict_template = dict()
    for map_name in map_name_and_houdini_parameter_name_dict.keys(): # less repeating code by generating it here
        maps_to_bake_dict_template[map_name] = False

    # recall that the benefit of class variables is that they aren't created for each instance all over again
    # + they can be accessed without instantiating the class


    def __init__(self, highpoly_path, lod_path, maps_to_bake_dict, bake_resolution_x, bake_resolution_y, export_directory, export_name_prefix = ""): # I haven't given a choice of export name, because that adds so much complexity
        # export_name_prefix is optional, and very worth it (a means to identify what you've baked other than the export_directory)
        # instead of making the Bake class tailored to megascans asset (which stops this from being a general thing)

        check_path(highpoly_path)
        self.highpoly_path = highpoly_path # e.g. "C:/User/highpoly.fbx"

        check_path(lod_path)
        self.lod_path = lod_path # e.g. "C:/User/lod.fbx"

        self.bake_resolution_tuple = (bake_resolution_x, bake_resolution_y)

        self.export_path = os_path_join_fix(export_directory, "{}_custom_baking_%(CHANNEL)s.rat".format(export_name_prefix)) # this is what the bake_texture node uses. Hardcoded along with export_name below

        # Setup map_name_and_export_paths_dict

        self.map_name_and_export_paths_dict = dict()
        for map_name in maps_to_bake_dict.keys():

            if maps_to_bake_dict[map_name] == True:
                parameter_name = self.map_name_and_houdini_parameter_name_dict[map_name]
                export_name = "{}_custom_baking_{}.exr".format(export_name_prefix, parameter_name.split("_")[-1]) # hardcoded to match self.export_path
                #^  parameter_name.split(" ")[-1], e.g. if the parameter name is 'vm_quickplane_Ds', the render token, %(CHANNEL)s, is 'Ds'
                self.map_name_and_export_paths_dict[map_name] = os_path_join_fix(export_directory, export_name)

        self.maps_to_bake_dict = maps_to_bake_dict




    def create_and_execute_in_houdini(self, housing_node): # includes executing the baking
        # Set up GEOs
        highpoly_geo_node = housing_node.createNode("geo", "Highpoly_geo_temp")
        lod_geo_node = housing_node.createNode("geo", "LOD_geo_temp") # aka lowpoly
        string_processor(highpoly_geo_node, "@cfile!file:{}".format(self.highpoly_path.replace("
```

```
", "%20")))
64          string_processor(lod_geo_node, "@cfile!file:{}".format(self.lod_path.replace(" ",
    "%20")))
65
66          # Set up camera
67          a_camera = housing_node.createNode("cam", "temp_camera")
68          string_processor(housing_node,
    "@etemp_camera!tx:int0!ty:int0!tz:int0!rx:int0!ry:int0!rz:int0!px:int0!py:int0!pz:int0!prx:int0!
    pry:int0!prz:int0!resx:int{}!resy:int{}".format(self.bake_resolution_tuple[0],
    self.bake_resolution_tuple[1])) # gross? perhaps set to default on t, r, p, pr is cleaner. Yep,
    definitely is.
69
70          # Set  up bake texture node
71          ropnet_node = housing_node.createNode("ropnet", "ropnet_for_baking")
72          baketexture_node = ropnet_node.createNode("baketexture::3.0", "bake_texture")
73          string_processor(ropnet_node, "@ebake_texture!camera:
    {}!vm_uvunwrapresx:int{}!vm_uvunwrapresy:int{}!vm_uvobject1:{}!vm_uvhires1:
    {}!vm_uvoutputpicture1:
    {}!vm_extractimageplanesformat:OpenEXR!vm_extractremoveintermediate:+!vm_uv_unwrap_method:int2".
    format(a_camera.path(), self.bake_resolution_tuple[0], self.bake_resolution_tuple[1],
    lod_geo_node.path(), highpoly_geo_node.path(), self.export_path.replace(" ", "%20"))) #TODO
74
75          # Iterate through maps_to_bake_dict, ticking parameters of corresponding maps which have
    True in the dict
76          for map_name in self.maps_to_bake_dict.keys():
77              parameter_name = self.map_name_and_houdini_parameter_name_dict[map_name]
78              corresponding_parm = baketexture_node.parm(parameter_name)
79
80              bake_bool = self.maps_to_bake_dict[map_name] # tr
81              if bake_bool == True:
82                  corresponding_parm.set(1) # set ticked
83              elif bake_bool == False:
84                  corresponding_parm.set(0) # set unticked
85              else:
86                  raise Exception("bake_bool: {}. Expected bake_bool to be
    boolean".format(bake_bool))
87
88          # Save and execute
89          hou.hipFile.save()
90          baketexture_node.parm("execute").pressButton()
91
92          return self.map_name_and_export_paths_dict # returning since it's new info (it wasn't
    passed in by the user)
93
94
95
96  class LOD:
97
98      def __init__(self, highpoly_path, polyreduce_percentage, export_path):
99          #check_path(highpoly_path)
100         self.highpoly_path = highpoly_path # e.g. "C:/User/geometry.fbx"
101
102         self.polyreduce_percentage = polyreduce_percentage
103
104         self.export_path = export_path
105
106
107     def create_and_execute_in_houdini(self, housing_node): # includes executing
108         custom_lod_node = housing_node.createNode("geo", "Custom_LOD")
109         string_processor(custom_lod_node,"cfile-file_node i0 cconvert-convert_node i0
    econvert_node i0 cpolyreduce::2.0-polyreduce_node i0 epolyreduce_node i0 crop_fbx-rop_fbx_node
    i0")
110
111         hou.hipFile.save() # save hip file before render
112         string_processor(custom_lod_node, "@efile_node!file:{} @epolyreduce_node!percentage:
    {}!reducepassedtarget:+!originalpoints:+ @erop_fbx_node!sopoutput:
    {}!execute:=".format(self.highpoly_path.replace(" ", "%20"), self.polyreduce_percentage,
    self.export_path.replace(" ", "%20")))
```