Posted on 2013年05月31日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 118 次

这个脚本可以将复杂的材质,比如有法线贴图的材质进行“烘焙”，转变为单一的贴图,可用来将Unity的游戏移植到移动平台时候使用。将这个脚本放Editor文件夹里，使用时选择一个Material材质，然后在菜单种“Custom/Bake Material”打开并调整照明和其他参数，点击Bake按钮，就会生成一个单一的贴图。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 001 | JavaScript - BakeMaterial.js |
| 002 |  |
| 003 | **class** BakeMaterialSettings |
| 004 |  |
| 005 | {&nbsp; Unity3D教程手册 |
| 006 |  |
| 007 | **private** **static** **var** kEditorPrefsName = “BakeMaterialSettings”; |
| 008 |  |
| 009 | **static** **var** kBakingLayerShouldBeUnusedInScene = 30; |
| 010 |  |
| 011 | **static** **var** kStandardTexNames = new Array (“\_MainTex”, “\_BumpMap”, “\_Detail”, “\_ParallaxMap”, “\_Parallax”); |
| 012 |  |
| 013 | **var** bakeAlpha = **false**; |
| 014 |  |
| 015 | **var** bakeMainTexAsWhite = **false**; |
| 016 |  |
| 017 | **var** minTextureResolution = 8; |
| 018 |  |
| 019 | **var** maxTextureResolution = 2048; |
| 020 |  |
| 021 | **var** emptyScene = **false**; |
| 022 |  |
| 023 | **var** useCustomLights = **false**; |
| 024 |  |
| 025 | **var** ambient = Color.black; |
| 026 |  |
| 027 | **static** **var** kLights = 3; |
| 028 |  |
| 029 | **var** enableLight = new boolean[kLights]; |
| 030 |  |
| 031 | **var** colorLight = new Color[kLights]; |
| 032 |  |
| 033 | **var** dirLight = new Vector2[kLights]; |
| 034 |  |
| 035 | function BakeMaterialSettings () |
| 036 |  |
| 037 | { |
| 038 |  |
| 039 | Load (); |
| 040 |  |
| 041 | } |
| 042 |  |
| 043 | function Load () |
| 044 |  |
| 045 | { |
| 046 |  |
| 047 | bakeAlpha = EditorPrefs.GetBool(kEditorPrefsName + “.bakeAlpha”); |
| 048 |  |
| 049 | bakeMainTexAsWhite = EditorPrefs.GetBool(kEditorPrefsName + “.bakeMainTexAsWhite”); |
| 050 |  |
| 051 | minTextureResolution = EditorPrefs.GetInt(kEditorPrefsName + “.minTextureResolution”, 8); |
| 052 |  |
| 053 | maxTextureResolution = EditorPrefs.GetInt(kEditorPrefsName + “.maxTextureResolution”, 2048); |
| 054 |  |
| 055 | emptyScene = EditorPrefs.GetBool(kEditorPrefsName + “.emptyScene”); |
| 056 |  |
| 057 | useCustomLights = EditorPrefs.GetBool(kEditorPrefsName + “.useCustomLights”); |
| 058 |  |
| 059 | ambient.r = EditorPrefs.GetFloat(kEditorPrefsName + “.ambient.r”); |
| 060 |  |
| 061 | ambient.g = EditorPrefs.GetFloat(kEditorPrefsName + “.ambient.g”); |
| 062 |  |
| 063 | ambient.b = EditorPrefs.GetFloat(kEditorPrefsName + “.ambient.b”); |
| 064 |  |
| 065 | ambient.a = EditorPrefs.GetFloat(kEditorPrefsName + “.ambient.a”, 1.0f); |
| 066 |  |
| 067 | **for** (**var** q = 0; q < kLights; ++q) |
| 068 |  |
| 069 | { |
| 070 |  |
| 071 | enableLight[q] = EditorPrefs.GetBool(kEditorPrefsName + “.enableLight” + q); |
| 072 |  |
| 073 | colorLight[q].r = EditorPrefs.GetFloat(kEditorPrefsName + “.color.r” + q, 0.5f); |
| 074 |  |
| 075 | colorLight[q].g = EditorPrefs.GetFloat(kEditorPrefsName + “.color.g” + q, 0.5f); |
| 076 |  |
| 077 | colorLight[q].b = EditorPrefs.GetFloat(kEditorPrefsName + “.color.b” + q, 0.5f); |
| 078 |  |
| 079 | colorLight[q].a = EditorPrefs.GetFloat(kEditorPrefsName + “.color.a” + q, 1.0f); |
| 080 |  |
| 081 | dirLight[q].x = EditorPrefs.GetFloat(kEditorPrefsName + “.dir.x” + q); |
| 082 |  |
| 083 | dirLight[q].y = EditorPrefs.GetFloat(kEditorPrefsName + “.dir.y” + q); |
| 084 |  |
| 085 | } |
| 086 |  |
| 087 | } |
| 088 |  |
| 089 | function Save () |
| 090 |  |
| 091 | { |
| 092 |  |
| 093 | EditorPrefs.SetBool(kEditorPrefsName + “.bakeAlpha”, bakeAlpha); |
| 094 |  |
| 095 | EditorPrefs.SetBool(kEditorPrefsName + “.bakeMainTexAsWhite”, bakeMainTexAsWhite); |
| 096 |  |
| 097 | EditorPrefs.SetInt(kEditorPrefsName + “.minTextureResolution”, minTextureResolution); |
| 098 |  |
| 099 | EditorPrefs.SetInt(kEditorPrefsName + “.maxTextureResolution”, maxTextureResolution); |
| 100 |  |
| 101 | EditorPrefs.GetBool(kEditorPrefsName + “.emptyScene”, emptyScene); |
| 102 |  |
| 103 | EditorPrefs.SetBool(kEditorPrefsName + “.useCustomLights”, useCustomLights); |
| 104 |  |
| 105 | EditorPrefs.SetFloat(kEditorPrefsName + “.ambient.r”, ambient.r); |
| 106 |  |
| 107 | EditorPrefs.SetFloat(kEditorPrefsName + “.ambient.g”, ambient.g); |
| 108 |  |
| 109 | EditorPrefs.SetFloat(kEditorPrefsName + “.ambient.b”, ambient.b); |
| 110 |  |
| 111 | EditorPrefs.SetFloat(kEditorPrefsName + “.ambient.a”, ambient.a); |
| 112 |  |
| 113 | **for** (**var** q = 0; q < kLights; ++q) |
| 114 |  |
| 115 | { |
| 116 |  |
| 117 | EditorPrefs.SetBool(kEditorPrefsName + “.enableLight” + q, enableLight[q]); |
| 118 |  |
| 119 | EditorPrefs.SetFloat(kEditorPrefsName + “.color.r” + q, colorLight[q].r); |
| 120 |  |
| 121 | EditorPrefs.SetFloat(kEditorPrefsName + “.color.g” + q, colorLight[q].g); |
| 122 |  |
| 123 | EditorPrefs.SetFloat(kEditorPrefsName + “.color.b” + q, colorLight[q].b); |
| 124 |  |
| 125 | EditorPrefs.SetFloat(kEditorPrefsName + “.color.a” + q, colorLight[q].a); |
| 126 |  |
| 127 | EditorPrefs.SetFloat(kEditorPrefsName + “.dir.x” + q, dirLight[q].x); |
| 128 |  |
| 129 | EditorPrefs.SetFloat(kEditorPrefsName + “.dir.y” + q, dirLight[q].y); |
| 130 |  |
| 131 | } |
| 132 |  |
| 133 | } |
| 134 |  |
| 135 | } |
| 136 |  |
| 137 | **class** BakeMaterial extends EditorWindow |
| 138 |  |
| 139 | { |
| 140 |  |
| 141 | **private** **static** **var** kMateriBakeNodeName = “\_\_MateriaBakeSetup”; |
| 142 |  |
| 143 | **private** **static** **var** kWindowMinSize = Vector2 (300, 386); |
| 144 |  |
| 145 | **private** **static** **var** settings : BakeMaterialSettings; |
| 146 |  |
| 147 | **private** **static** **var** visible : boolean = **false**; |
| 148 |  |
| 149 | **private** **var** camera : GameObject; |
| 150 |  |
| 151 | **private** **var** plane : GameObject; |
| 152 |  |
| 153 | **private** **var** previewTexture : Texture; |
| 154 |  |
| 155 | **private** **var** lights : GameObject[] = new GameObject[BakeMaterialSettings.kLights]; |
| 156 |  |
| 157 | **private** **var** stateChanged = **false**; |
| 158 |  |
| 159 | **private** **var** texViewScrollPosition = Vector2.zero; |
| 160 |  |
| 161 | **private** **var** lastMaterial : Material; |
| 162 |  |
| 163 | **private** **var** originalScene = “”; |
| 164 |  |
| 165 | **private** **var** scheduleBakeOnNextUpdate = **false**; |
| 166 |  |
| 167 | **private** function SetupScene () |
| 168 |  |
| 169 | { |
| 170 |  |
| 171 | DestroyScene (); |
| 172 |  |
| 173 | **var** oldGo = GameObject.Find(kMateriBakeNodeName); |
| 174 |  |
| 175 | **if** (oldGo) |
| 176 |  |
| 177 | DestroyImmediate (oldGo); |
| 178 |  |
| 179 | camera = new GameObject (kMateriBakeNodeName, Camera); |
| 180 |  |
| 181 | plane = GameObject.CreatePrimitive (PrimitiveType.Plane); |
| 182 |  |
| 183 | **var** cam = camera; |
| 184 |  |
| 185 | cam.camera.backgroundColor = Color.black; |
| 186 |  |
| 187 | cam.camera.clearFlags = CameraClearFlags.SolidColor; |
| 188 |  |
| 189 | cam.camera.orthographic = **true**; |
| 190 |  |
| 191 | cam.camera.orthographicSize = 5.0; |
| 192 |  |
| 193 | cam.camera.cullingMask = 1 << settings.kBakingLayerShouldBeUnusedInScene; |
| 194 |  |
| 195 | plane.transform.parent = cam.transform; |
| 196 |  |
| 197 | plane.transform.position = Vector3.forward \* 10.0; |
| 198 |  |
| 199 | plane.transform.rotation = Quaternion.Euler (0, 0, 180) \* Quaternion.Euler (-90, 0, 0); |
| 200 |  |
| 201 | plane.layer = settings.kBakingLayerShouldBeUnusedInScene; |
| 202 |  |
| 203 | **for** (**var** l **in** lights) |
| 204 |  |
| 205 | { |
| 206 |  |
| 207 | l = new GameObject (“Light”, Light); |
| 208 |  |
| 209 | l.light.type = LightType.Directional; |
| 210 |  |
| 211 | l.light.cullingMask = 1 << settings.kBakingLayerShouldBeUnusedInScene; |
| 212 |  |
| 213 | l.transform.parent = cam.transform; |
| 214 |  |
| 215 | l.active = **false**; |
| 216 |  |
| 217 | } |
| 218 |  |
| 219 | } |
| 220 |  |
| 221 | **private** function UpdateScene (m : Material) |
| 222 |  |
| 223 | { |
| 224 |  |
| 225 | **for** (q = 0; q < settings.kLights; ++q) |
| 226 |  |
| 227 | { |
| 228 |  |
| 229 | lights[q].active = settings.useCustomLights & settings.enableLight[q]; |
| 230 |  |
| 231 | lights[q].light.color = settings.colorLight[q]; |
| 232 |  |
| 233 | lights[q].transform.rotation = |
| 234 |  |
| 235 | Quaternion.AngleAxis(settings.dirLight[q].x, Vector3.up) \* |
| 236 |  |
| 237 | Quaternion.AngleAxis(settings.dirLight[q].y, Vector3.right); |
| 238 |  |
| 239 | } |
| 240 |  |
| 241 | **if** (settings.useCustomLights) |
| 242 |  |
| 243 | RenderSettings.ambientLight = settings.ambient; |
| 244 |  |
| 245 | **else** **if** (settings.emptyScene) |
| 246 |  |
| 247 | RenderSettings.ambientLight = Color.white; |
| 248 |  |
| 249 | plane.renderer.material = m; |
| 250 |  |
| 251 | } |
| 252 |  |
| 253 | **private** function DestroyScene () |
| 254 |  |
| 255 | { |
| 256 |  |
| 257 | GameObject.DestroyImmediate (camera); |
| 258 |  |
| 259 | GameObject.DestroyImmediate (plane); |
| 260 |  |
| 261 | GameObject.DestroyImmediate (previewTexture); |
| 262 |  |
| 263 | } |
| 264 |  |
| 265 | function UpdateMaterialPreview (m : Material) : RenderTexture |
| 266 |  |
| 267 | { |
| 268 |  |
| 269 | **if** (!m) |
| 270 |  |
| 271 | **return**; |
| 272 |  |
| 273 | **var** saveAmbientLight = RenderSettings.ambientLight; |
| 274 |  |
| 275 | **var** saveMainTexture = m.mainTexture; |
| 276 |  |
| 277 | **if** (settings.bakeMainTexAsWhite) |
| 278 |  |
| 279 | m.mainTexture = **null**; |
| 280 |  |
| 281 | *// setup* |
| 282 |  |
| 283 | **if** (!camera) |
| 284 |  |
| 285 | SetupScene (); |
| 286 |  |
| 287 | camera.SetActiveRecursively(**true**); |
| 288 |  |
| 289 | UpdateScene (m); |
| 290 |  |
| 291 | **var** res = FindLargestTextureResolution (plane.renderer.sharedMaterial, settings.minTextureResolution, settings.maxTextureResolution); |
| 292 |  |
| 293 | **var** rt = RenderCameraToRenderTexture (camera.camera, res.x, res.y); |
| 294 |  |
| 295 | *// restore* |
| 296 |  |
| 297 | camera.SetActiveRecursively(**false**); |
| 298 |  |
| 299 | RenderSettings.ambientLight = saveAmbientLight; |
| 300 |  |
| 301 | m.mainTexture = saveMainTexture; |
| 302 |  |
| 303 | previewTexture = rt; |
| 304 |  |
| 305 | **return** rt; |
| 306 |  |
| 307 | } |
| 308 |  |
| 309 | function CaptureMaterial(m : Material) |
| 310 |  |
| 311 | { |
| 312 |  |
| 313 | **var** matAssetPath = AssetDatabase.GetAssetPath (m); |
| 314 |  |
| 315 | **var** assetPath = System.IO.Path.Combine (System.IO.Path.GetDirectoryName (matAssetPath), System.IO.Path.GetFileNameWithoutExtension (matAssetPath)); |
| 316 |  |
| 317 | **var** rt = UpdateMaterialPreview (m); |
| 318 |  |
| 319 | RenderTextureToPNG (rt, settings.bakeAlpha, assetPath + “.png”); |
| 320 |  |
| 321 | } |
| 322 |  |
| 323 | function OnEnable () |
| 324 |  |
| 325 | { |
| 326 |  |
| 327 | **if** (!settings) |
| 328 |  |
| 329 | settings = new BakeMaterialSettings (); |
| 330 |  |
| 331 | SetupScene (); |
| 332 |  |
| 333 | visible = **true**; |
| 334 |  |
| 335 | } |
| 336 |  |
| 337 | function OnDisable () |
| 338 |  |
| 339 | { |
| 340 |  |
| 341 | DestroyScene (); |
| 342 |  |
| 343 | settings.Save (); |
| 344 |  |
| 345 | visible = **false**; |
| 346 |  |
| 347 | } |
| 348 |  |
| 349 | **static** function GetTargetMaterial () : Material |
| 350 |  |
| 351 | { |
| 352 |  |
| 353 | **return** EditorUtility.InstanceIDToObject (Selection.activeInstanceID) **as** Material; |
| 354 |  |
| 355 | } |
| 356 |  |
| 357 | function OnSelectionChange () |
| 358 |  |
| 359 | { |
| 360 |  |
| 361 | Repaint (); |
| 362 |  |
| 363 | } |
| 364 |  |
| 365 | function Update () |
| 366 |  |
| 367 | { |
| 368 |  |
| 369 | **var** rebuildScene = **false**; |
| 370 |  |
| 371 | **if** (scheduleBakeOnNextUpdate) |
| 372 |  |
| 373 | { |
| 374 |  |
| 375 | Bake (); |
| 376 |  |
| 377 | scheduleBakeOnNextUpdate = **false**; |
| 378 |  |
| 379 | rebuildScene = **true**; |
| 380 |  |
| 381 | } |
| 382 |  |
| 383 | **if** (originalScene == “” && EditorApplication.currentScene == “”) |
| 384 |  |
| 385 | settings.emptyScene = **true**; |
| 386 |  |
| 387 | **if** (settings.emptyScene && originalScene == “” && EditorApplication.currentScene != “”) |
| 388 |  |
| 389 | { |
| 390 |  |
| 391 | DestroyScene (); |
| 392 |  |
| 393 | **if** (EditorApplication.SaveCurrentSceneIfUserWantsTo ()) |
| 394 |  |
| 395 | { |
| 396 |  |
| 397 | originalScene = EditorApplication.currentScene; |
| 398 |  |
| 399 | EditorApplication.NewScene (); |
| 400 |  |
| 401 | } |
| 402 |  |
| 403 | **else** |
| 404 |  |
| 405 | settings.emptyScene = **false**; |
| 406 |  |
| 407 | rebuildScene = **true**; |
| 408 |  |
| 409 | } |
| 410 |  |
| 411 | **else** **if** (!settings.emptyScene && originalScene != “”) |
| 412 |  |
| 413 | { |
| 414 |  |
| 415 | EditorApplication.OpenScene (originalScene); |
| 416 |  |
| 417 | rebuildScene = **true**; |
| 418 |  |
| 419 | originalScene = “”; |
| 420 |  |
| 421 | } |
| 422 |  |
| 423 | **if** (rebuildScene) |
| 424 |  |
| 425 | { |
| 426 |  |
| 427 | SetupScene (); |
| 428 |  |
| 429 | } |
| 430 |  |
| 431 | **if** (rebuildScene || stateChanged || !settings.emptyScene) |
| 432 |  |
| 433 | { |
| 434 |  |
| 435 | UpdateMaterialPreview (lastMaterial); |
| 436 |  |
| 437 | Repaint (); |
| 438 |  |
| 439 | stateChanged = **false**; |
| 440 |  |
| 441 | } |
| 442 |  |
| 443 | } |
| 444 |  |
| 445 | function OnGUI () |
| 446 |  |
| 447 | { |
| 448 |  |
| 449 | **var** material = GetTargetMaterial (); |
| 450 |  |
| 451 | **if** (lastMaterial != material) |
| 452 |  |
| 453 | UpdateMaterialPreview (material); |
| 454 |  |
| 455 | **if** (material) |
| 456 |  |
| 457 | lastMaterial = material; |
| 458 |  |
| 459 | EditorGUILayout.BeginHorizontal(); |
| 460 |  |
| 461 | EditorGUILayout.BeginVertical(GUILayout.MaxWidth(200)); |
| 462 |  |
| 463 | **if** (!(originalScene == “” && EditorApplication.currentScene == “”)) |
| 464 |  |
| 465 | { |
| 466 |  |
| 467 | settings.emptyScene = !EditorGUILayout.BeginToggleGroup(“Scene ligthing”, !settings.emptyScene); |
| 468 |  |
| 469 | EditorGUILayout.EndToggleGroup(); |
| 470 |  |
| 471 | } |
| 472 |  |
| 473 | settings.useCustomLights = EditorGUILayout.BeginToggleGroup(“Custom lighting”, settings.useCustomLights); |
| 474 |  |
| 475 | **if** (settings.useCustomLights) |
| 476 |  |
| 477 | { |
| 478 |  |
| 479 | EditorGUI.indentLevel = 1; |
| 480 |  |
| 481 | settings.ambient = EditorGUILayout.ColorField(“Ambient”, settings.ambient); |
| 482 |  |
| 483 | **for** (**var** q = 0; q < settings.kLights; ++q) |
| 484 |  |
| 485 | { |
| 486 |  |
| 487 | settings.enableLight[q] = EditorGUILayout.BeginToggleGroup(“Light”, settings.enableLight[q]); |
| 488 |  |
| 489 | EditorGUI.indentLevel = 2; |
| 490 |  |
| 491 | settings.colorLight[q] = EditorGUILayout.ColorField(“Color”, settings.colorLight[q]); |
| 492 |  |
| 493 | settings.dirLight[q] = EditorGUILayout.Vector2Field(“Direction”, settings.dirLight[q]); |
| 494 |  |
| 495 | EditorGUILayout.EndToggleGroup(); |
| 496 |  |
| 497 | } |
| 498 |  |
| 499 | } |
| 500 |  |
| 501 | EditorGUI.indentLevel = 0; |
| 502 |  |
| 503 | EditorGUILayout.EndToggleGroup(); |
| 504 |  |
| 505 | settings.bakeAlpha = EditorGUILayout.Toggle(“Bake Alpha”, settings.bakeAlpha); |
| 506 |  |
| 507 | settings.bakeMainTexAsWhite = !EditorGUILayout.Toggle(“MainTex”, !settings.bakeMainTexAsWhite); |
| 508 |  |
| 509 | settings.minTextureResolution = EditorGUILayout.IntField(“Min Resolution”, settings.minTextureResolution); |
| 510 |  |
| 511 | settings.maxTextureResolution = EditorGUILayout.IntField(“Max Resolution”, settings.maxTextureResolution); |
| 512 |  |
| 513 | settings.minTextureResolution = Mathf.Max(2, settings.minTextureResolution); |
| 514 |  |
| 515 | settings.maxTextureResolution = Mathf.Max(settings.minTextureResolution, settings.maxTextureResolution); |
| 516 |  |
| 517 | EditorGUILayout.BeginHorizontal(); |
| 518 |  |
| 519 | **if** (GUILayout.Button(“Bake”)) |
| 520 |  |
| 521 | { |
| 522 |  |
| 523 | CaptureMaterial (lastMaterial); |
| 524 |  |
| 525 | } |
| 526 |  |
| 527 | **if** (GUILayout.Button(“Bake Selected”)) |
| 528 |  |
| 529 | { |
| 530 |  |
| 531 | scheduleBakeOnNextUpdate = **true**; |
| 532 |  |
| 533 | } |
| 534 |  |
| 535 | EditorGUILayout.EndHorizontal(); |
| 536 |  |
| 537 | EditorGUILayout.EndVertical(); |
| 538 |  |
| 539 | texViewScrollPosition = EditorGUILayout.BeginScrollView (texViewScrollPosition); |
| 540 |  |
| 541 | **var** r = GUILayoutUtility.GetAspectRect(1.0f); |
| 542 |  |
| 543 | **if** (previewTexture) |
| 544 |  |
| 545 | EditorGUI.DrawPreviewTexture(r, previewTexture); |
| 546 |  |
| 547 | EditorGUILayout.EndScrollView(); |
| 548 |  |
| 549 | EditorGUILayout.EndHorizontal(); |
| 550 |  |
| 551 | **if** (GUI.changed) |
| 552 |  |
| 553 | { |
| 554 |  |
| 555 | stateChanged = **true**; |
| 556 |  |
| 557 | } |
| 558 |  |
| 559 | } |
| 560 |  |
| 561 | @MenuItem(“Custom/Bake Material ...”, **false**, 5) |
| 562 |  |
| 563 | **static** function CreateBakeEditor() |
| 564 |  |
| 565 | { |
| 566 |  |
| 567 | **var** window = EditorWindow.GetWindow(BakeMaterial); |
| 568 |  |
| 569 | window.title = “Bake Material”; |
| 570 |  |
| 571 | window.minSize = kWindowMinSize; |
| 572 |  |
| 573 | window.Show(); |
| 574 |  |
| 575 | } |
| 576 |  |
| 577 | @MenuItem(“Custom/Bake Selected Materials”, **false**, 4) |
| 578 |  |
| 579 | **static** function Bake() |
| 580 |  |
| 581 | { |
| 582 |  |
| 583 | **var** instanceIDs = Selection.instanceIDs; |
| 584 |  |
| 585 | **var** currentScene = EditorApplication.currentScene; |
| 586 |  |
| 587 | **var** wasAlreadyVisible = BakeMaterial.visible; |
| 588 |  |
| 589 | **var** window = EditorWindow.GetWindow(BakeMaterial); |
| 590 |  |
| 591 | **if** (window.settings.emptyScene) |
| 592 |  |
| 593 | { |
| 594 |  |
| 595 | **if** (!EditorApplication.SaveCurrentSceneIfUserWantsTo ()) |
| 596 |  |
| 597 | **return**; |
| 598 |  |
| 599 | EditorApplication.NewScene (); |
| 600 |  |
| 601 | } |
| 602 |  |
| 603 | window.SetupScene (); |
| 604 |  |
| 605 | **for** (**var** i **in** instanceIDs) |
| 606 |  |
| 607 | { |
| 608 |  |
| 609 | **var** m : Material = EditorUtility.InstanceIDToObject (i) **as** Material; |
| 610 |  |
| 611 | **if** (m) |
| 612 |  |
| 613 | window.CaptureMaterial (m); |
| 614 |  |
| 615 | } |
| 616 |  |
| 617 | window.DestroyScene (); |
| 618 |  |
| 619 | **if** (window.settings.emptyScene && currentScene) |
| 620 |  |
| 621 | { |
| 622 |  |
| 623 | EditorApplication.OpenScene (currentScene); |
| 624 |  |
| 625 | } |
| 626 |  |
| 627 | **if** (!wasAlreadyVisible) |
| 628 |  |
| 629 | window.Close (); |
| 630 |  |
| 631 | } |
| 632 |  |
| 633 | **static** function FindLargestTextureResolution (m : Material, minTexRes : **int**, maxTexRes : **int**) : Vector2 |
| 634 |  |
| 635 | { |
| 636 |  |
| 637 | **var** res = Vector2 (minTexRes, minTexRes); |
| 638 |  |
| 639 | **for** (**var** n **in** BakeMaterialSettings.kStandardTexNames) |
| 640 |  |
| 641 | { |
| 642 |  |
| 643 | **if** (!m.HasProperty (n)) |
| 644 |  |
| 645 | **continue**; |
| 646 |  |
| 647 | **var** t : Texture = m.GetTexture (n); |
| 648 |  |
| 649 | **if** (!t) |
| 650 |  |
| 651 | **continue**; |
| 652 |  |
| 653 | res.x = Mathf.Max (res.x, t.width); |
| 654 |  |
| 655 | res.y = Mathf.Max (res.y, t.height); |
| 656 |  |
| 657 | } |
| 658 |  |
| 659 | res.x = Mathf.Min (res.x, maxTexRes); |
| 660 |  |
| 661 | res.y = Mathf.Min (res.y, maxTexRes); |
| 662 |  |
| 663 | **return** res; |
| 664 |  |
| 665 | } |
| 666 |  |
| 667 | **static** function RenderCameraToRenderTexture (cam : Camera, width : **int**, height : **int**) : RenderTexture |
| 668 |  |
| 669 | { |
| 670 |  |
| 671 | **var** rt = cam.camera.targetTexture; |
| 672 |  |
| 673 | **if** (rt && rt.width != width && rt.height != height) |
| 674 |  |
| 675 | DestroyImmediate(rt); |
| 676 |  |
| 677 | **if** (!rt) |
| 678 |  |
| 679 | rt = new RenderTexture (width, height, 24); |
| 680 |  |
| 681 | cam.camera.targetTexture = rt; |
| 682 |  |
| 683 | cam.camera.Render (); |
| 684 |  |
| 685 | **return** rt; |
| 686 |  |
| 687 | } |
| 688 |  |
| 689 | **static** function RenderTextureToPNG (rt : RenderTexture, bakeAlpha : boolean, assetPath : **String**) |
| 690 |  |
| 691 | { |
| 692 |  |
| 693 | RenderTexture.active = rt; |
| 694 |  |
| 695 | **var** screenShot = new Texture2D (rt.width, rt.height, bakeAlpha? TextureFormat.ARGB32 : TextureFormat.RGB24, **false**); |
| 696 |  |
| 697 | screenShot.ReadPixels (Rect (0, 0, rt.width, rt.height), 0, 0); |
| 698 |  |
| 699 | RenderTexture.active = **null**; |
| 700 |  |
| 701 | **var** bytes = screenShot.EncodeToPNG (); |
| 702 |  |
| 703 | System.IO.File.WriteAllBytes (assetPath, bytes); |
| 704 |  |
| 705 | AssetDatabase.ImportAsset (assetPath, ImportAssetOptions.ForceUpdate); |
| 706 |  |
| 707 | } |
| 708 |  |
| 709 | } |