




















Tutorial de incluir um novo boneco no jogo:

1) baixar os arquivos .obj da internet na pasta data

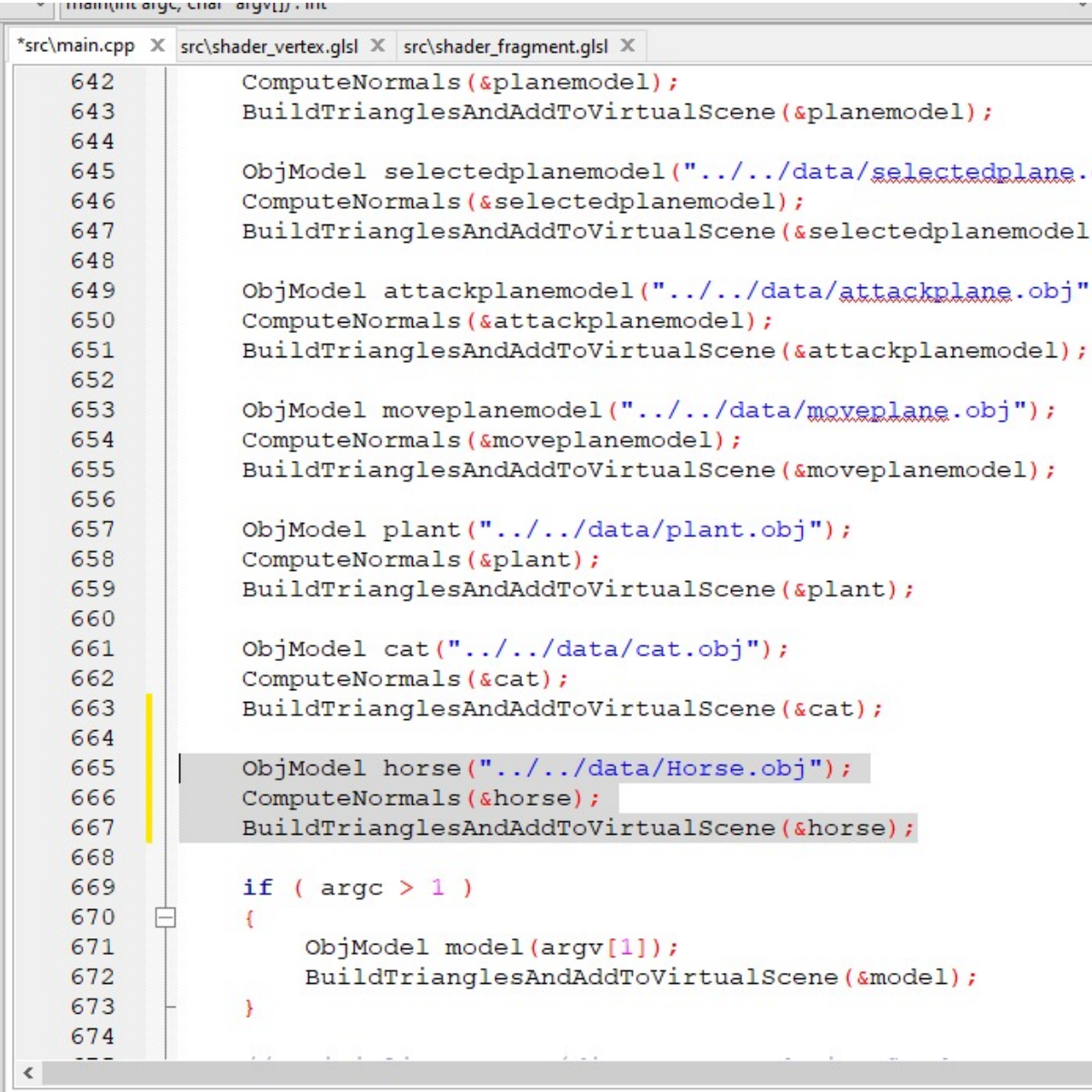
Meu Computador > Área de Trabalho > THALES FACUL > FCG > Trab2020-2 > Trab-FCG > data

Nome	Data de modificação	Tipo	Tamanho
 attack	23/12/2020 02:51	Arquivo JPG	23 KB
 attackplane	23/12/2020 05:17	3D Object	1 KB
 bunny	26/10/2020 09:01	3D Object	2.443 KB
 cat	20/12/2011 07:12	3D Object	5.247 KB
 horse texture	09/04/2006 20:16	Arquivo JPG	123 KB
 Horse	17/08/2018 02:44	3D Object	966 KB
 move	23/12/2020 04:12	Arquivo JPG	22 KB
 moveplane	23/12/2020 05:17	3D Object	1 KB
 plane	26/10/2020 09:01	3D Object	1 KB
 plant	03/01/2021 15:40	3D Object	66.484 KB
 selected	22/12/2020 22:46	Arquivo JPG	22 KB
 selected	22/12/2020 05:16	Arquivo PNG	5 KB
 selectedplane	22/12/2020 23:04	3D Object	1 KB
 sphere	26/10/2020 09:01	3D Object	15 KB
 tc-earth_daymap_surface	26/10/2020 09:01	Arquivo JPG	658 KB
 tc-earth_nightmap_citylights	26/10/2020 09:01	Arquivo GIF	329 KB
 tree_lima	03/01/2021 16:22	Arquivo JPG	7.078 KB
 unselected	22/12/2020 05:29	Arquivo JPG	14 KB
 V3 BrownEyes 12	25/09/2003 07:22	Arquivo JPG	52 KB

- 2) Criar o personagem criando uma instância do objeto nas declarações iniciais de characters (atentar ao “HORSE” em maiúsculas seguindo o padrão que nem nos outros) e incluir no vetor de chars do stage1 (pré definidos).
Atentar pra posição dos tilesArray ser uma que está livre e existente.

```
src/main.cpp x src/shader_vertex.glsl x src/shader_fragment.glsl x
516 bool checkIfCharIsOnTile(int tileID);
517 void clearAllMovableTiles();
518 void clearAllAttackableTiles();
519 void computeAttack(int attacker, int defender);
520 void levelUpChar(int charIndex);
521
522 //TextRendering - Custom Functions
523 void TextRendering_TileDetails(GLFWwindow* window);
524 void TextRendering_CharacterDetails(GLFWwindow* window);
525
526 //Stages
527 Stage stage1 = stage1Creation();
528
529 //Characters
530 Character bunnyCitizen=newCharacter(0,stage1.tilesArray[3].id,"Bunny Citizen",1,Citizen,"BUNNY","Normal");
531 Character bunnyWarrior=newCharacter(1,stage1.tilesArray[10].id,"Bunny Warrior",1,Warrior,"BUNNY","Dark");
532 Character bunnyCleric=newCharacter(2,stage1.tilesArray[16].id,"Bunny Cleric",1,Priest,"BUNNY","Light");
533 Character bunnyAnimal=newCharacter(3,stage1.tilesArray[32].id,"Bunny",1,Animal,"BUNNY","Normal");
534 Character plant1=newCharacter(4,stage1.tilesArray[1].id,"Plant",1,Plant,"PLANT","Grass");
535 Character catAnimal=newCharacter(5,stage1.tilesArray[40].id,"Cat",1,Animal,"CAT","Normal");
536 Character horseAnimal=newCharacter(6,stage1.tilesArray[42].id,"Horse",1,Animal,"HORSE","Dark");
537
538 std::vector<Character> stage1Characters = {bunnyCitizen,bunnyWarrior,bunnyCleric,bunnyAnimal,plant1,catAnimal,horseAnimal};
539
540 //constantes
541 int selectedTile;
542 int stage1lastID;
543 bool movingAction=false;
544 int movingCharacterID;
545 bool attackingAction=false;
546 int attackingCharacterID=-1;
547 bool engaging=false;
548 int defenderID;
```

3) criar o obj model do mesmo jeito que os outros obj são criados (easy peasy)



```
main(argc, char *argv[]) {
src/main.cpp X src/shader_vertex.glsl X src/shader_fragment.glsl X
642     ComputeNormals(&planemodel);
643     BuildTrianglesAndAddToVirtualScene(&planemodel);
644
645     ObjModel selectedplanemodel("../data/selectedplane..
646     ComputeNormals(&selectedplanemodel);
647     BuildTrianglesAndAddToVirtualScene(&selectedplanemodel
648
649     ObjModel attackplanemodel("../data/attackplane.obj"
650     ComputeNormals(&attackplanemodel);
651     BuildTrianglesAndAddToVirtualScene(&attackplanemodel);
652
653     ObjModel moveplanemodel("../data/moveplane.obj");
654     ComputeNormals(&moveplanemodel);
655     BuildTrianglesAndAddToVirtualScene(&moveplanemodel);
656
657     ObjModel plant("../data/plant.obj");
658     ComputeNormals(&plant);
659     BuildTrianglesAndAddToVirtualScene(&plant);
660
661     ObjModel cat("../data/cat.obj");
662     ComputeNormals(&cat);
663     BuildTrianglesAndAddToVirtualScene(&cat);
664
665     ObjModel horse("../data/Horse.obj");
666     ComputeNormals(&horse);
667     BuildTrianglesAndAddToVirtualScene(&horse);
668
669     if ( argc > 1 )
670     {
671         ObjModel model(argv[1]);
672         BuildTrianglesAndAddToVirtualScene(&model);
673     }
674
675 }
```

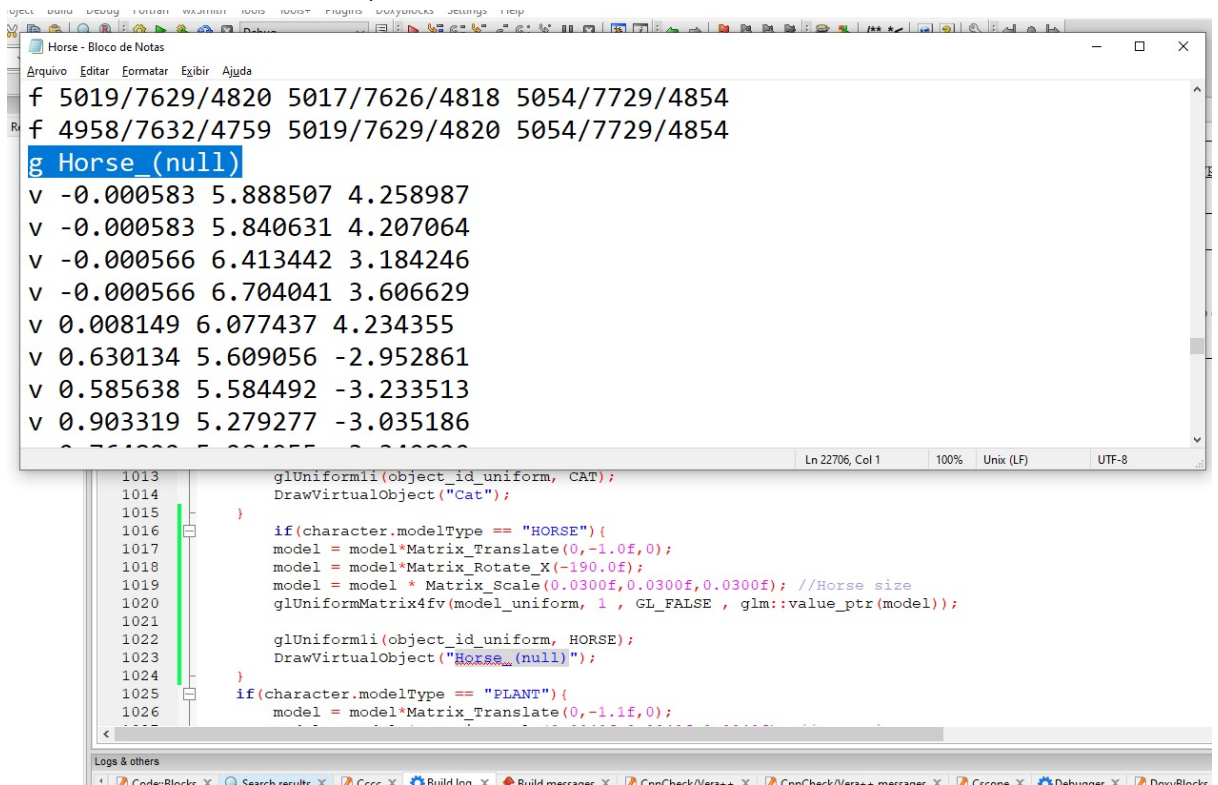
- 4) No arquivo shader fragment definir o número do personagem seguindo a ordem

```
src/main.cpp X src/shader_vertex.glsl X src/shader_fragment.glsl X
7   in vec4 position_world;
8   in vec4 normal;
9
10  // Posição do vértice atual no sistema de coo
11  in vec4 position_model;
12
13  // Coordenadas de textura obtidas do arquivo
14  in vec2 texcoords;
15
16  // Matrizes computadas no código C++ e enviad
17  uniform mat4 model;
18  uniform mat4 view;
19  uniform mat4 projection;
20
21  // Identificador que define qual objeto está
22  #define SPHERE 0
23  #define BUNNY 1
24  #define PLANE 2
25  #define SELECTEDPLANE 3
26  #define ATTACKPLANE 4
27  #define MOVEPLANE 5
28  #define PLANT 6
29  #define CAT 7
30  #define HORSE 8
31
32  uniform int object_id;
33
34  // Parâmetros da axis-aligned bounding box (A
35  uniform vec4 bbox_min;
36  uniform vec4 bbox_max;
```

- 5) Ainda no shader fragment, incluir um if para o personagem criado (a princípio a definição de color será igual aos outros já existentes, mas pode variar)

```
src\main.cpp X src\shader_vertex.glsl X src\shader_fragment.glsl X
151
152     color = Kd0 * (lamBERT + 0.01);
153
154     if(object_id == PLANE){
155         color = Kd2 * (lamBERT + 0.01);
156     }
157     if(object_id == SELECTEDPLANE){
158         color = Kd3 * (lamBERT + 0.01);
159     }
160     if(object_id == ATTACKPLANE){
161         color = Kd4 * (lamBERT + 0.01);
162     }
163     if(object_id == MOVEPLANE){
164         color = Kd5 * (lamBERT + 0.01);
165     }
166
167     if(object_id == PLANT){
168         color = Kd6 * (lamBERT + 0.01);
169     }
170
171     if(object_id == HORSE){
172         color = Kd6 * (lamBERT + 0.01);
173     }
174
175     if(object_id == CAT){
```


- 6) Por fim , na função de drawCharacter definir a posição, rotação (se necessário) e scale do modelo que nem os outros, aqui talvez seja necessário ir arrumando na mão o tamanho do scale e da rotação até ficar bonitinho, testando no jogo. Importante que o nome definido no drawVirtualObject seja igual ao definido depois de g dentro do arquivo .obj (só abrir com o bloco de notas e dar ctrl f, se tiver mais de um testa todos até dar certo)



The screenshot shows a code editor with two windows. The top window, titled 'Horse - Bloco de Notas', contains an OBJ file for a horse model. The bottom window shows C++ code in a function that draws a character, with a line for drawing the horse model highlighted in blue.

```
f 5019/7629/4820 5017/7626/4818 5054/7729/4854
f 4958/7632/4759 5019/7629/4820 5054/7729/4854
g Horse_(null)
v -0.000583 5.888507 4.258987
v -0.000583 5.840631 4.207064
v -0.000566 6.413442 3.184246
v -0.000566 6.704041 3.606629
v 0.008149 6.077437 4.234355
v 0.630134 5.609056 -2.952861
v 0.585638 5.584492 -3.233513
v 0.903319 5.279277 -3.035186
v 0.751000 5.000000 0.000000

1013 glUniformli(object_id_uniform, CAT);
1014 DrawVirtualObject("Cat");
1015 }
1016 if(character.modelType == "HORSE"){
1017     model = model*Matrix_Translate(0,-1.0f,0);
1018     model = model*Matrix_Rotate_X(-190.0f);
1019     model = model * Matrix_Scale(0.0300f,0.0300f,0.0300f); //Horse size
1020     glUniformMatrix4fv(model_uniform, 1, GL_FALSE, glm::value_ptr(model));
1021
1022     glUniformli(object_id_uniform, HORSE);
1023     DrawVirtualObject("Horse_(null)");
1024 }
1025 if(character.modelType == "PLANT"){
1026     model = model*Matrix_Translate(0,-1.1f,0);
```

7) Só rodar e testar

