**O Menino do Morro**

**Projeto de TCC**

**Game Design Document: Tecnológico**

**Integrantes:**

**Henrique Valerio   RA00117247**

**Juan Villegas     RA00111916**

|  |
| --- |
| **Sumário** |
|  |

|  |
| --- |
| **1- Histórico do Projeto** |
| Iniciamos o projeto com o protótipo que possuia somente movimentação e morte do personagem. Com o avanço do desenvolvimento, inserimos os cenários e desenvolvemos os inimigos. A partir daí, nos concentramos nas mecânicas do jogo, como tiro e o controle de animações. Com esta parte concluída, passamos a desenvolver o boss e seu funcionamento, concluindo assim a versão para lançamento. |

|  |
| --- |
| **2 – Interface** |
| **2.1 – Sistema Visual**  **6.1.1 – HUD(Head-Up Display)**  Munição e armas e pontuação.  **6.1.2 – Menus**  No Menu o jogador terá acesso a tela de “Voltar ao Jogo”, “Opções”(som), “Tutorial” e “Sair”.  **6.1.3 – Sistema de Renderização**  Para aqueles que utilizam o sistema operacional Microsoft Windows©, será usado o DirectX(9c).  **6.1.4 – Câmera**  Somente Top View.  **2.2 – Sistema de Controle**  Movimentação - W,A,S,D  Mira - Mover o mouse  Atirar - Botão esquerdo do mouse  Pegue munição - Botão direito do mouse  Pausa/Menu - Esc  **2.3 – Sistema de Áudio**  **6.3.1 – Músicas**  Estéreo  **6.3.2 – Efeitos sonoros**  Estéreo  **2.4 – Sistema de Ajuda**  Na opção “Tutorial” do menu do game. |

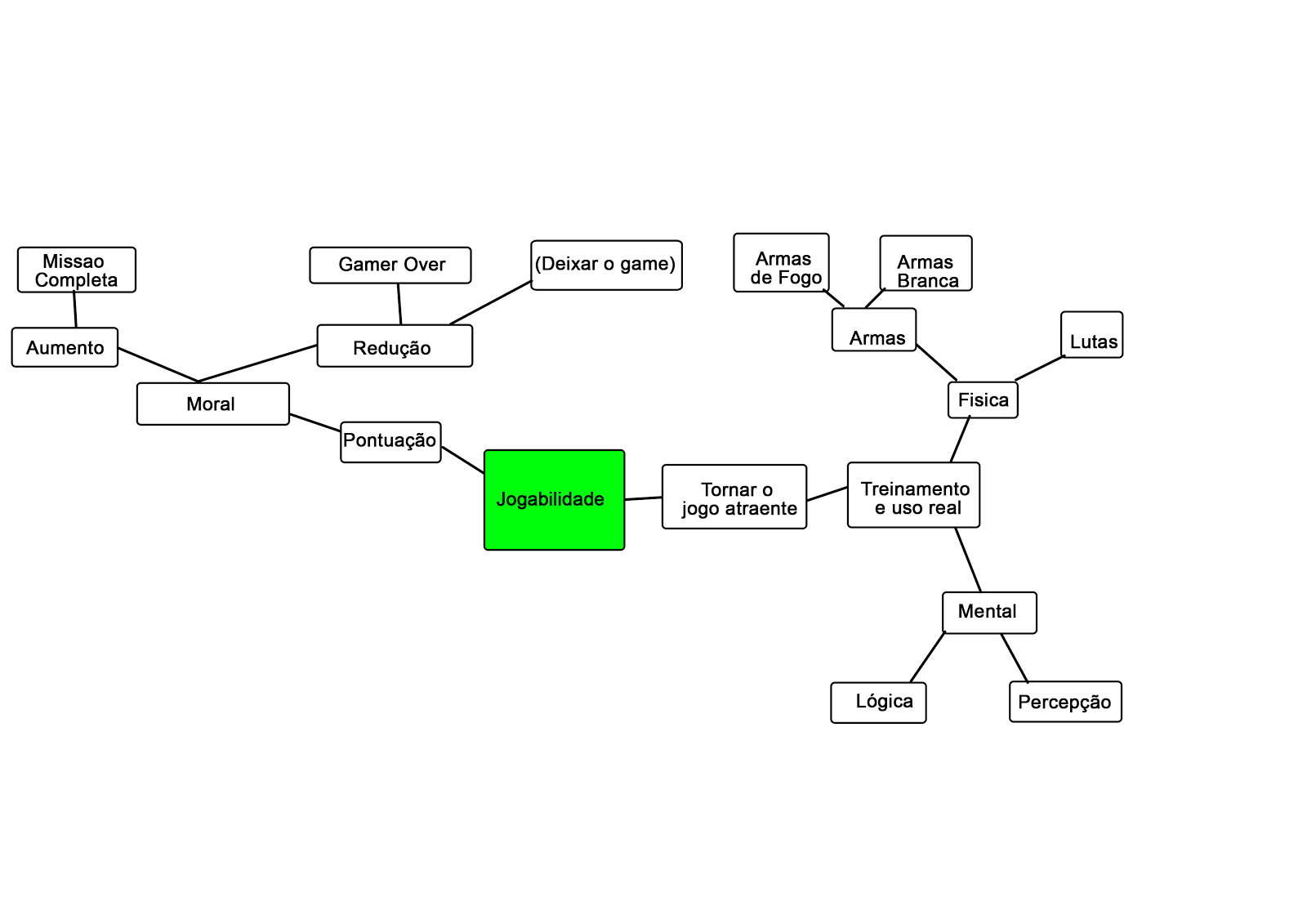
|  |
| --- |
| **3 – Inteligência Artificial** |
| **3.1 – IA de Oponentes**  Inimigos estarão em suas áreas e atacarão o personagem à primeira vista. Se atingido o personagem morre.  **3.2 – IA de Inimigos**  Inimigos permanecerão em suas áreas atacando o protagonista quando o virem. O boss seguirá o mesmo padrão, mas precisará de mais acertos para morrer.  **3.3 – Personagens Não-Combatentes**  Não haverá personagens não combatentes, além dos amigáveis que participarão do game apenas em cutscenes.  **3.4 – Personagens Amigáveis**  Os personagens amigáveis aparecerão apenas em cutscenes.  **3.5 – IA de suporte**  **3.5.1 – Colisões do jogador e objetos**  O jogador colidirá com objetos no cenário que impedirão sua passagem. Serão inseridas portas para que o personagem entre em determinados locais.  **3.5.2 – Melhor caminho (Pathfinding)**  O único meio de prosseguir é eliminando todos os inimigos em suas respectivas áreas. O jogador deve escolher onde atacar primeirio. |

|  |
| --- |
| **4 – Projeto Técnico** |
| **4.1 – Equipamento-alvo**  O game será produzido para a plataforma PC e os requisitos mínimos:  Sistema Operacional: Windows XP/7 (x86 e x64)  Processador: 1.2GHz  RAM: 512 MB ​​RAM  Placa de Vídeo: Placa gráfica DirectX 9 compatível com pelo menos 32 MB de memória de vídeo  **4.2 – Ambiente desenvolvido** (Hardware e Software)  Será produzido nos computadores da Faculdade, usando o Windows 7 e Unity versão 4.5.2.  **4.3 – Procedimentos e padrões de Desenvolvimento**  **4.4 – Motor do Jogo (Engine)**  Unity versão (4.5.2)  **4.5 – Rede**  Interação com a rede será apenas para compra e download.  **4.6 – Linguagem de programação**  Utilização do MonoDevelop na linguagem C#.  **4.6.1 – audio.cs**  using UnityEngine;  using System.Collections;  [ExecuteInEditMode]  public class audio : MonoBehaviour {  // DECLARA E INICIALIZA VARIAVEIS  public float volumeSlider = 10.0F; // tamanho da barra de volume  static float volume; // variavel de volume  public bool menuInicial;  void OnGUI()  {  if (menuInicial) { // SE FOR SLIDER DO MENU INICIAL  volumeSlider = GUI.HorizontalSlider (new Rect (Screen.width / 2 + 210, Screen.height / 2 + 90, 100, 5), volumeSlider, 0.0F, 1.0F); // posicao da barra, tamanho da barra e valores de inicio/fim  AudioListener.volume = volumeSlider; //volume igual a 0.1 vezes a posicao da barra  }  else // SE FOR SLIDER DO MENU INGAME  {  volumeSlider = GUI.HorizontalSlider (new Rect (Screen.width / 2 + 210, Screen.height / 2 - 45, 100, 5), volumeSlider, 0.0F, 1.0F); // posicao da barra, tamanho da barra e valores de inicio/fim  AudioListener.volume = volumeSlider; //volume igual a 0.1 vezes a posicao da barra  }  }  }  **4.6.2 – cenasCarregar.cs**  using UnityEngine;  using System.Collections;  public class cenasCarregar : MonoBehaviour {  // DECLARACAO E INICIALIZACAO DE VARIAVEIS  private string nomedacena = "1-1 Loading"; // nome da cena a ser carregada  private string menu = "menu";  private string pont = "10 Pont";  public string faseJogada = "3 Cidade";  public bool novoMenu;  public bool loadMenu;  public bool sairMenu;  public bool novoInGame;  public bool sairInGame;  public bool novoPont;  public bool sairPont;      void Awake ()  {  faseJogada = PlayerPrefs.GetString ("ChaveUltimaFase", "3 Cidade");  }    // AO CLICAR NO BOTAO SETADO COMO NOVOMENU, SEU SAVE SERA RESETADO E O JOGO SERIA INICIADO  void OnMouseDown ()  {  // AO CLICAR NO BOTAO SETADO COMO NOVOMENU, SEU SAVE SERA RESETADO E O JOGO SERIA INICIADO DESDE O COMECO  if (novoMenu){  PlayerPrefs.DeleteKey ("ChaveSalvarPontos");  PlayerPrefs.DeleteKey ("ChaveInimigosMortos");  PlayerPrefs.DeleteKey("ChaveUltimaFase");  Application.LoadLevel(nomedacena);  }  // AO CLICAR NO BOTAO SETADO COMO LOADMENU, SEU JOGO SERA INICIADO A PARTIR DO ULTIMO PONTO SALVO  if (loadMenu) {  Application.LoadLevel (faseJogada);  }  // AO CLICAR NO BOTAO SETADO COMO SAIRMENU, O JOGO SERA ENCERRADO  if (sairMenu) {  Application.Quit();  }  // AO CLICAR NO BOTAO SETADO COMO NOVOINGAME, SEU JOGO SERA INICIADO A PARTIR DO ULTIMO PONTO SALVO  if (novoInGame) {  Application.LoadLevel(faseJogada);  }  // AO CLICAR NO BOTAO SETADO COMO SAIRINGAME, SEU JOGO MOSTRARA A TELA DE PONTUACAO COM OS ULTIMOS DADOS SALVOS  if (sairInGame) {  Application.LoadLevel (pont);  }  // AO CLICAR NO BOTAO SETADO COMO NOVOPONT, SEU JOGO SERA CARREGADO A PARTIR DO ULTIMO PONTO SALVO  if (novoPont) {  Application.LoadLevel (faseJogada);  }  // AO CLICAR NO BOTAO SETADO COMO SAIRPONT, O JOGO CARREGARA O MENU  if (sairPont){  Application.LoadLevel (menu);  }  }  }  **4.6.3 – clicavelInMenu.cs**  using UnityEngine;  using System.Collections;  public class clicavelInMenu : MonoBehaviour {    public int newsortingOrder = 1; // nova posicao na layer  public GameObject[] listaTelasEsconder; // lista de telas a esconder  public GameObject mostrarPlayer;  public float tempoAtivar = 0.5f;      // Update is called once per frame  void OnMouseDown ()  {  Time.timeScale=1; // velocidade do tempo  mostrarPlayer.SetActive (true);  foreach (GameObject s in listaTelasEsconder) { // para cada gameobject na lista a esconder    (s.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = -1; // mandar o sprite para a ordem -1 da layer  s.SetActive (false); // tornar objeto inativo  }  }  }  **4.6.4 – ingame.cs**  using UnityEngine;  using System.Collections;  public class ingame : MonoBehaviour {    public int newsortingOrder = 1; // nova posicao na layer  public GameObject[] listaTelasEsconder; // lista de telas a esconder  public GameObject telaMostrar; // tela que deve ser mostrada  public GameObject mostrarPlayer;    // Update is called once per frame  void Update () {  if (Input.GetKey (KeyCode.Escape)) { // se a tecla ESC for pressionada  //foreach (rotate in gameObject.  Time.timeScale = 0.0f; // velocidade do tempo  mostrarPlayer.SetActive (false);  foreach (GameObject s in listaTelasEsconder) { // para cada gameobject na lista a esconder    (s.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = -1; // mandar o sprite para a ordem -1 da layer  s.SetActive (false); // tornar objeto inativo  }  if (telaMostrar != null) // se a lista a mostrar nao estiver vazia  (telaMostrar.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = newsortingOrder; // mandar o objeto a mostrar para a nova ordem  telaMostrar.SetActive (true); // tornar este objeto ativo  }  }  }  **4.6.5 – menu.cs**  using UnityEngine;  using System.Collections;  public class menu : MonoBehaviour  {  public int newsortingOrder = 1; // nova posicao na layer  public GameObject[] listaTelasEsconder; // lista de telas a esconder  public GameObject telaMostrar; // tela que deve ser mostrada      void OnMouseDown ()  {  foreach (GameObject s in listaTelasEsconder) { // para cada gameobject na lista a esconder    (s.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = -1; // mandar o sprite para a ordem -1 da layer  s.SetActive (false); // tornar objeto inativo  }  if (telaMostrar != null) // se a lista a mostrar nao estiver vazia  (telaMostrar.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = newsortingOrder; // mandar o objeto a mostrar para a nova ordem  telaMostrar.SetActive (true); // tornar este objeto ativo    }  }  **4.6.6 – Administrador\_Botoes.cs**  using UnityEngine;  using System.Collections;  public class Administrador\_Botoes : MonoBehaviour  {  private GameObject[] ArrayBotoes;  //public BoxCollider2D[] BotoesParaDesativar;  // Use this for initialization  // Update is called once per frame    public void Troca\_Botoes (string NomeDoBotaoSelecionado)  {  ArrayBotoes = GameObject.FindGameObjectsWithTag("BotaoDoMenu"); // incrementa arraybotoes com objetos usando a tag BotaoDoMenu  foreach(GameObject bot in ArrayBotoes)  {  Botoes botAtual = bot.GetComponent<Botoes>()as Botoes; //altera botao atual  botAtual.BarraNegra.SetActive(false); // desativa objeto  botAtual.Tela.SetActive(false); // desativa objeto  botAtual.Selecionado = false; // desativa objeto  botAtual.enabled = false; // desativa objeto  }  Botoes botaoASerAtivado = GameObject.Find (NomeDoBotaoSelecionado).GetComponent<Botoes> () as Botoes; // encontra objetos com script Botoes  botaoASerAtivado.Selecionado = true; // exibir objeto  botaoASerAtivado.BarraNegra.SetActive (true); // exibir objeto  botaoASerAtivado.Tela.SetActive (true); // exibir objeto  botaoASerAtivado.enabled = true; // exibir objeto  }  public void Troca\_Tela()  {  }  }  **4.6.7 – AIWayPointsCS .cs**  using UnityEngine;  using System.Collections;  public class AIWayPointsCS : MonoBehaviour {  // DECLARACAO DE VARIAVEIS  public Transform wayPointCorrente = null;  public float velocidade = 1.0f; // velocidade de movimento  public GameObject jogador ;  public float distanciaCalculada ;  public float anguloCalculado ;  public float distanciaMinima = 2.0f;  public float anguloMinimo = 15.0f;  public bool moveParaJogador = false;  public bool enxergaJogador = false;  public LayerMask layerMaskEnemy ;  public Animator animator ;  public bool ignorarJogador = false;  public bool e1hands ;  public bool e1pistol ;  public bool e1smg ;  public bool e1shotgun ;  public bool e2hands ;  public bool e2pistol ;  public bool e2smg ;  public bool e2shotgun ;  public float fireRate = 0.5f;  private float nextFire = 0.0f;      public void Start() {    jogador = GameObject.FindGameObjectWithTag("Player");  animator = this.gameObject.GetComponentInChildren<Animator>();  }    // tela de pontuacao vai grapontos, inimigos mortos e a fase anterior numa string  public void Update () {    //Debug.Log("Nome inimigo: " + gameObject.name + " enxerga: " + enxergaJogador);    float distanciaCalculada = Vector3.Distance(jogador.transform.position,transform.position);  float anguloCalculado = Vector3.Angle(jogador.transform.position,transform.position);  // MOVER-SE EM DIRECAO DESDE QUE A DISTANCIA E ANGULO ESTEJAM DENTRO DOS LIMITES, NPC DEVE OLHAR EM DIRECAO AO JOGADOR  if(ignorarJogador == false) {  if (distanciaCalculada <= distanciaMinima)  {  if (enxergaJogador)  {  if (anguloCalculado <= anguloMinimo)  {  moveParaJogador = true;  transform.position = Vector3.MoveTowards(transform.position,jogador.transform.position,velocidade \* Time.deltaTime);  transform.LookAt(jogador.transform.position);  } else {  moveParaJogador = false;  }    } else {  moveParaJogador = false;  }  } else {  moveParaJogador = false;  }      // SE INIMIGO1 ESTIVER DESARMADO, A PARTIR DESTA DISTANCIA TOCAR ANIMACAO E AUDIO DE SOCO  /\*\*/ if (e1hands)  {  if (moveParaJogador && distanciaCalculada < 0.4f)  {  animator.Play("Inimigo1 - Bare Shoot");  audio.Play();  }  }  // SE INIMIGO2 ESTIVER DESARMADO, A PARTIR DESTA DISTANCIA TOCAR ANIMACAO E AUDIO DE SOCO  /\*\*/ if (e2hands)  {  if (moveParaJogador && distanciaCalculada < 0.4f)  {  animator.Play("Inimigo2 - Bare Shoot");  audio.Play();  }  }  // SE INIMIGO1 ESTIVER ARMADO COM UMA PISTOLA, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e1pistol)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo1 - Pistol Shoot");  audio.Play();  RaycastHit2D hit1 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit1.collider != null) {  if(hit1.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  //Debug.DrawLine (transform.position, hit.point, Color.red);  }  }  // SE INIMIGO2 ESTIVER ARMADO COM UMA PISTOLA, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e2pistol)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo2 - Pistol Shoot");  audio.Play();    RaycastHit2D hit4 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit4.collider != null) {  if(hit4.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  //Debug.DrawLine (transform.position, hit.point, Color.red);  }  }  // SE INIMIGO1 ESTIVER ARMADO COM UMA SMG, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e1smg)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo1 - SMG Shoot");  audio.Play();    RaycastHit2D hit2 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit2.collider != null) {  if(hit2.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  }  }  // SE INIMIGO2 ESTIVER ARMADO COM UMA SMG, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e2smg)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo2 - SMG Shoot");  audio.Play();    RaycastHit2D hit5 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit5.collider != null) {  if(hit5.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  }  }  // SE INIMIGO1 ESTIVER ARMADO COM UMA SHOTGUN, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e1shotgun)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo1 - Shotgun Shoot");  audio.Play();    RaycastHit2D hit6 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit6.collider != null) {  if(hit6.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  }  }  // SE INIMIGO2 ESTIVER ARMADO COM UMA SHOTGUN, A PARTIR DESTA DISTANCIA E COM O TEMPO DEFINIDO, TOCAR ANIMACAO E AUDIO DE TIRO E DISPARAR EM DIRECAO AO JOGADOR  /\*\*/ else if (e2shotgun)  {  if (moveParaJogador && distanciaCalculada < 2.0f && Time.time > nextFire)  {  nextFire = Time.time + fireRate;  animator.Play("Inimigo2 - Shotgun Shoot");  audio.Play();    RaycastHit2D hit3 = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy);  if(hit3.collider != null) {  if(hit3.collider.gameObject.tag.Equals("Player")) {  jogador.GetComponent<morte>().jogadorMorre();  }  } else {  enxergaJogador = false;  }  }  }      if(wayPointCorrente != null && moveParaJogador == false) { // se o waypoint atual nao for vazio    transform.position = Vector3.MoveTowards(transform.position,wayPointCorrente.position,velocidade \* Time.deltaTime); // mover em direcao ao proximo waypoint na velocidade definida  transform.LookAt(wayPointCorrente); // olhar em direcao ao waypoint enquanto se move em direcao a ele  }  } else {  if(wayPointCorrente != null) { // se o waypoint atual nao for vazio  transform.position = Vector3.MoveTowards(transform.position,wayPointCorrente.position,velocidade \* Time.deltaTime); // mover em direcao ao proximo waypoint na velocidade definida  transform.LookAt(wayPointCorrente); // olhar em direcao ao waypoint enquanto se move em direcao a ele  }  }    }  // NPC ANDA EM LINHA RETA ATE SEU PROXIMO WAYPOINT, A NAO SER QUE PLAYER ENTRE EM SEU CAMPO DE VISAO, ESTE ENTAO SE TORNA SEU PROXIMO WAYPOINT  public void FixedUpdate () {  RaycastHit2D informaCaoColisao = Physics2D.Linecast(transform.position,jogador.transform.position,layerMaskEnemy);  //Debug.Log(informaCaoColisao.collider.tag);  if(informaCaoColisao.collider != null) {  if(informaCaoColisao.collider.tag.Equals("Player")) {  enxergaJogador = true;  } else {  enxergaJogador = false;  }  }  }  // AO COLIDIR CM O JOGADOR, JOGADOR MORRE  public void OnCollisionEnter2D(Collision2D colisorPlayer) {  if (colisorPlayer.gameObject.tag == "Player")  {  jogador.GetComponent<morte>().jogadorMorre();  moveParaJogador = false;  enxergaJogador = false;  }  }  // IGNORAR JOGADOR  public void IgnorarPlayer() {  ignorarJogador = true;  }  // AO TOCAR EM UM TRIGGER (WAYPOINT), O PROXIMO WAYPOINT VINCULADO A ESTE SERA O PROXIMO WAYPOINT DO NPC  public void OnTriggerEnter2D(Collider2D objeto) {  if(objeto.tag.Equals("Waypoint")) { // se a tag do objeto colidido for waypoint  ProximosWaypointsCS proximosWaypoints = objeto.GetComponent<ProximosWaypointsCS>(); // define possiveis waypoints  Transform proximoWayPoint = proximosWaypoints.ObterProximaPosicao(); // alterar o proximo waypoint  // Calculo de tempo  //velocidade = Vector3.Distance(proximoWayPoint.position,gameObject.transform.position)/tempoDeslocamento;  wayPointCorrente = proximoWayPoint; // waypoint atual volta para a lista de proximos waypoints  }  }  }  **4.6.8 – ammo.cs**  using UnityEngine;  using System.Collections;  public class Ammo : MonoBehaviour {    public GameObject pistola;  public bool onTrigger = false;  Collider2D outroColisor2;  void Update ()  {  if (Input.GetButtonDown ("Interact") && onTrigger) // DENTRO DO TRIGGER PRESSIONE O BOTAO INTERACT  {  audio.Play(); // TOCA AUDIO  outroColisor2.gameObject.GetComponentInChildren<Weapon>().getWeapon(); // OBTEM SCRIPT Weapon NavMeshObstacle FILHO do COLISOR  Component spriteAmmo = GetComponent<SpriteRenderer>(); // SPRITE DA ARMA  Destroy(spriteAmmo); // DESTRUIR SPRITE  Component colliderAmmo = GetComponent<BoxCollider2D>(); // COLLIDER  Destroy(colliderAmmo); // DESTRUIR COLLIDER  onTrigger = false; // PLAYER PERDE EFEITO SOBRE OBJETO  }    }    void OnTriggerEnter2D (Collider2D outroColisor) // AO ENTRAR NO TRIGGER, PLAYER TERA ACAO  {  if (outroColisor.gameObject.tag == "Player")  {  onTrigger=true;  outroColisor2 = outroColisor;    }    }    void OnTriggerExit2D (Collider2D outroColisor) // AO SAIR DO TRIGGER, NAO TERA MAIS EFEITO  {  if (outroColisor.gameObject.tag == "Player")  {  onTrigger=false;    }    }    }  **4.6.9 – ammoCounter.cs**  using UnityEngine;  using System.Collections;  public class ammoCounter : MonoBehaviour {  public Weapon muni;    // CONTADOR DE MUNICAO A SER EXIBIDO NA TELA  void Awake ()  {  muni = GameObject.FindObjectOfType <Weapon> ();  }    void Update () {  guiText.text = "" + muni.pistolAmmo;  }  }  **4.6.2 – BossAI.cs**  using UnityEngine;  using System.Collections;  public class BossAI : MonoBehaviour {  // DECLARACAO DE VARIAVEIS  public Animator animator;  public AudioClip audioAtirando;  public AudioClip audioRecarregando;  public GameObject jogador;  public LayerMask layerMaskEnemy;  public float tempoDisparo = 0.5f;  public float tempoRecarregar = 1.5f;  public int estadoBoss = 1; // 1= carregando e 2= atirando  public float contadorTempo = 0.0f;  public bool ignorarJogador = false;      void Start () {  jogador = GameObject.FindGameObjectWithTag("Player"); // variavel jogador sera composta pelo objeto com tag player  animator = this.gameObject.GetComponentInChildren<Animator>(); // animator sera o animator filho deste objeto  audio.clip = audioRecarregando; // valor inicial do audio  audio.Play(); // toca audio  }        void Update () {  transform.LookAt(jogador.transform.position); // olhar para jogador  contadorTempo = Time.deltaTime + contadorTempo; // contador de tempo  RaycastHit2D hit = Physics2D.Linecast(transform.position, jogador.transform.position, layerMaskEnemy); // disparo em direcao ao jogador  if(hit.collider != null && estadoBoss == 2) { // colisor diferente de vazio e boss atirando  if(hit.collider.gameObject.tag.Equals("Player")) { // colidir com tag player  jogador.SendMessage("jogadorMorre"); // jogador morre  }    }      if(contadorTempo >= tempoRecarregar && estadoBoss == 1) { // contador e estado do boss como condicoes  contadorTempo = 0; // reseta contador  animator.Play("Matheus - Shoot"); // animacao atirando  audio.Stop(); // para o audio anterior  audio.clip = audioAtirando; // muda faixa de audio  audio.Play(); // toca audio atual  audio.loop = true; // toca audio em loop  estadoBoss = 2; // muda estado do boss para atirando    }  if(contadorTempo >= tempoDisparo && estadoBoss == 2) { // contador e estado do boss como condicoes  contadorTempo = 0; // reseta contador  audio.Stop(); // para o audio anterior  audio.clip = audioRecarregando; // muda faixa de audio  audio.Play(); // toca audio atual  animator.Play("Matheus - Carregando");    estadoBoss = 1; // muda estado do boss para carregando  }    if (ignorarJogador == true) // se ignorar jogador  {  animator.Play("Matheus - Idle"); // animacao idle  audio.Stop(); // para o audio  }    }        public void IgnorarPlayer() { // funcao usada pelo script morte do jogador  ignorarJogador = true; // ignora jogador  }    }  **4.6.2 – botoes.cs**  using UnityEngine;  using System.Collections;  public class Botoes : Administrador\_Botoes  {  // DECLARACAO E INICIALIZACAO DE VARIAVEIS  public GameObject  Tela,  BarraNegra;  public bool Selecionado;  public Administrador\_Botoes \_administradorBotoes;  public void Start()  {  \_administradorBotoes = GameObject.Find ("Administrador de Botões").GetComponent<Administrador\_Botoes> ();  BarraNegra.SetActive(false);  Tela.SetActive (false);  Debug.Log (this.gameObject.name);  DesativarBotoesInternos ();  }  // AO COLOCAR O MOUSE SOBRE O BOTAO, UMA TARJA PRETA ACENDERA SOB O BOTAO  void OnMouseOver()  {  BarraNegra.SetActive (true);  }  // AO RETIRAR O MOUSE DO BOTAO, A TARJA PRETA SE APAGARA SOB O BOTAO SE O BOTAO NAO ESTIVER CLICADO  void OnMouseExit()  {  if(Selecionado)  BarraNegra.SetActive (true);  else if(!Selecionado)  BarraNegra.SetActive(false);  }  // SE DETERMINADO BOTAO FOR CLICADO, OUTRA TELA IRA SURGIR, JUNTO COM SEUS NOVOS BOTOES, ENQUANTO DESABILITA OUTROS  void OnMouseDown()  {  \_administradorBotoes.Troca\_Botoes ((string)this.gameObject.name);  if ((string)this.gameObject.name == "Botao Creditos" || (string)this.gameObject.name == "BotãoCreditos TrocaTela Esq" || (string)this.gameObject.name == "BotãoCreditos TrocaTela Dir")  {  BoxCollider2D[] BCInternos = GameObject.Find("Botão Créditos 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternos)  boxes.enabled = true;  }  else  {  BoxCollider2D[] BCInternos = GameObject.Find("Botão Créditos 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternos)  boxes.enabled = false;  }  if ((string)this.gameObject.name == "Botao Tutorial" || (string)this.gameObject.name == "BotãoTuto TrocaTela Esq" || (string)this.gameObject.name == "BotãoTuto TrocaTela Dir")  {  BoxCollider2D[] BCInternos = GameObject.Find("Botao Tutorial 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternos)  boxes.enabled = true;  }  else  {  BoxCollider2D[] BCInternos = GameObject.Find("Botao Tutorial 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternos)  boxes.enabled = false;  }  }  // AO HABILITAR BOTOES COM CLIQUES, OUTROS INACESSIVEIS NO MOMENTO FICAM BLOQUEADOS  void DesativarBotoesInternos()  {  BoxCollider2D[] BCInternosCreditos = GameObject.Find("Botão Créditos 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternosCreditos)  {  boxes.enabled = false;  }    BoxCollider2D[] BCInternosTuto = GameObject.Find("Botao Tutorial 2").GetComponentsInChildren<BoxCollider2D>();  foreach(BoxCollider2D boxes in BCInternosTuto)  {  boxes.enabled = false;  }  }  }  **4.6.2 – cameraFollow.cs**  using UnityEngine;  using System.Collections;  public class CameraFollow : MonoBehaviour  {  public float velocidade = 20 ; // velocidade da camera  public Transform siga; // o que deve ser seguido    public void SetarAlvo(Transform t)  {  siga = t; // valor local do que deve ser seguido para que a câmera sempre comece no personagem  transform.position = new Vector3 (t.position.x, t.position.y, transform.position.z); // posicao do que deve ser seguido    }  void LateUpdate()  {  if(siga) // se houver algo no siga  {  float x = IncrementarDirecao(transform.position.x, siga.position.x, velocidade);  float y = IncrementarDirecao(transform.position.y, siga.position.y, velocidade);  transform.position = new Vector3(x, y, transform.position.z);  // move a camera em direcao ao objeto definido como siga com a velocidade definida  }  }  private float IncrementarDirecao (float VelocidadeAt, float VelocidadeAl, float aceler)  {  if(VelocidadeAt == VelocidadeAl) // se velocidade atual for igual a velocidade do alvo  {  return VelocidadeAt; // retornar a velocidade atual  }  else  {  float direcao = Mathf.Sign(VelocidadeAl - VelocidadeAt); // direcao na velocidade atual subtraida da velocidade do alvo - ao se aproximar desacelera  VelocidadeAt += aceler \* Time.smoothDeltaTime \* direcao; // VelocidadeAt deve ser aumentada ou diminuida para se aproximar do alvo  return (direcao == Mathf.Sign(VelocidadeAl - VelocidadeAt))? VelocidadeAt: VelocidadeAl; // Se VelocidadeAt agora passou da VelocidadeAl então retornar VelocidadeAlv, senão retornar VelocidadeAt  }  }  }  **4.6.2 – ChangeLevel.cs**  using UnityEngine;  using System.Collections;  public class ChangeLevel : MonoBehaviour {    public string nomeDaCena = "nome da cena"; // cena a ser carregada  public Weapon pontosSalvos; // chama script weapon    void Start () {  pontosSalvos = GameObject.FindObjectOfType<Weapon> ();  }    void OnCollisionEnter2D(Collision2D outroColisor) // se outro colisor tocar este colisor  {  if (outroColisor.gameObject.tag == "Player") { // se o colisor for marcado como Player, salvar valores de ChaveInimigosMortos, ChaveSalvarPontos e ChaveUltimaFase  PlayerPrefs.SetInt ("ChaveSalvarPontos", pontosSalvos.pontuacao);  PlayerPrefs.SetInt ("ChaveInimigosMortos", pontosSalvos.inimigosMortos);  PlayerPrefs.SetString("ChaveUltimaFase", nomeDaCena);  Application.LoadLevel (nomeDaCena); // carregar a cena definida  }  }  }  **4.6.2 – ControleAnim.cs**  using UnityEngine;  using System.Collections;  public class ControleAnim : MonoBehaviour {  // DECLARACAO E INICIALIZACAO DE VARIAVEIS  private Animator animator;  private HashIDs hash;  void Awake ()  {  animator = GetComponent<Animator> ();  hash = gameObject.GetComponent<HashIDs> ();  }    void FixedUpdate ()  {  // SE O PERSONAGEM SE MOVIMENTAR HORIZONTAL OU VERTICALMENTE, ATIVAR ANIMACAO DE MOVIMENTO  if (Mathf.Abs (Input.GetAxis ("Horizontal")) > 0.1 || Mathf.Abs (Input.GetAxis ("Vertical")) > 0.1f)  animator.SetBool (hash.walkBool, true);  else if (Mathf.Abs (Input.GetAxis ("Horizontal")) < 0.1 || Mathf.Abs (Input.GetAxis ("Vertical")) < 0.1f)  animator.SetBool (hash.walkBool, false);  }  }  **4.6.2 – EnemyCounter.cs**  using UnityEngine;  using System.Collections;  public class EnemyCounter : MonoBehaviour {    // DECLARACAO E INICIALIZACAO DE VARIAVEIS    public GameObject[] enemies;  int enemiesLeft;  ArrayList inimigos;  public GameObject[] listaMostrar;  public bool cenarioFinal = false;      void Start ()  {  inimigos = new ArrayList ();  GameObject[] enemies = GameObject.FindGameObjectsWithTag("Enemy");  enemiesLeft = enemies.Length;  foreach (GameObject obj in enemies) {  inimigos.Add(obj.GetComponent<EnemyScript>());  }  }    void Update () {  // PARA CADA OBJETO QUE POSSUA UM ENEMYSCRIPT E ENQUANTO NAO ESTIVER MORTO, ADICIONE MAIS UM A LISTA DE INIMIGOS  enemiesLeft = 0;  if (inimigos != null) {  foreach (EnemyScript s in inimigos)  if (!s.mateInimigo)  enemiesLeft++;    // SE O NUMERO DE INIMIGOS FOR IGUAL A ZERO E NAO FOR O CENARIO FINAL, HABILITAR OS OBJETOS  if (enemiesLeft == 0 && cenarioFinal == false) {  foreach (GameObject obj in listaMostrar) { // para cada objeto listado a ser escondido  obj.SetActive (true);    }  }  }  }  }  **4.6.2 – EnemyScript.cs**  using UnityEngine;  using System.Collections;  public class EnemyScript : MonoBehaviour {  // DECLARACAO E INICIALIZACAO DE VARIAVEIS  public Animator animator = new Animator(); // inicia um animator  private string animFinal = "9-1 AnimFinal";  public bool mateInimigo = false;  public bool e1;  public bool e2;  public bool boss;  public int BossHealth = 5;  public int pontosDerrota = 0;  public SpriteRenderer SRInimigo;  public string layerNameDead = "Morto";  public GameObject ammo;  public Weapon savedScore;  void Start () {  savedScore = GameObject.FindObjectOfType<Weapon> ();  }  // MATAR INIMIGO A SER CHAMADO DE OUTRO SCRIPT  public void MatarInimigo()  {  mateInimigo = true;    }  public void Update() {    // SE FOR INIMIGO2, TOCAR ANIMACAO DE MORTE, DESATIVAR ESTES COMPONENTES E MANDAR O SPRITE PARA A LAYER DEFINIDA  if (e2)  {  if (mateInimigo) {  animator.Play ("Inimigo2 - Dying"); // tocar a animacao Inimigo2 - Dead  Component objAI = GetComponent("AIWayPointsCS");  Component objAI2 = GetComponent<CircleCollider2D>();  Destroy(objAI);  Destroy(objAI2);  Destroy(GetComponent<EnemyScript>());  SRInimigo.sortingLayerName = layerNameDead;  if (ammo != null){  ammo.SetActive (true);}  }    }  // SE FOR INIMIGO1, TOCAR ANIMACAO DE MORTE, DESATIVAR ESTES COMPONENTES E MANDAR O SPRITE PARA A LAYER DEFINIDA  if (e1)  {  if (mateInimigo) {  animator.Play ("Inimigo1 - Dying"); // tocar a animacao Inimigo1 - Dead  Component objAI = GetComponent ("AIWayPointsCS");  Component objAI2 = GetComponent<CircleCollider2D> ();  Destroy (objAI);  Destroy (objAI2);  Destroy (GetComponent<EnemyScript> ());  SRInimigo.sortingLayerName = layerNameDead;  if (ammo != null){  ammo.SetActive (true);}  }  }  // SE FOR O BOSS, REDUZIR HP DE UM EM UM ATE QUE SEU HP ZERE E ENTAO A ANIMACAO FINAL SERA EXIBIDA  if (boss)  {  if (mateInimigo) {  BossHealth = BossHealth - 1;  mateInimigo = false;  }  if (BossHealth == 0)  {  // PlayerPrefs.SetInt ("ChaveSalvarPontos", savedScore.pontuacao);  // PlayerPrefs.SetInt ("ChaveInimigosMortos", savedScore.inimigosMortos);  Application.LoadLevel(animFinal);  }  }  }  }  **4.6.2 – HashIDs.cs**  using UnityEngine;  using System.Collections;  public class HashIDs : MonoBehaviour {  // DECLARA E INICIALIZA HASHS PARA FACILITAR A DEFINICAO NOS ANIMATORS  public int deadState;  public int deadBool;  public int walkState;  public int walkBool;  public int shootState;  // public int shootBool;  public int idleState;  public int shootTrigger;  // public int walkTrigger;  void Awake ()  {  deadState = Animator.StringToHash ("Base Layer.Dead");  deadBool = Animator.StringToHash ("deadBool");  walkState = Animator.StringToHash ("Base Layer.PistolWalk");  walkBool = Animator.StringToHash ("walkBool");  shootState = Animator.StringToHash ("Base Layer.PistolShoot");  // shootBool = Animator.StringToHash ("shootBool");  idleState = Animator.StringToHash ("Base Layer.PistolIdle");  shootTrigger = Animator.StringToHash ("shootTrigger");  // walkTrigger = Animator.StringToHash ("walkTrigger");  }    }  **4.6.2 – matheusAnim.cs**  using UnityEngine;  using System.Collections;  public class matheusAnim : MonoBehaviour {    public float timeLeft = 3.0f; // tempo para o objeto desaparecer  public GameObject[] listaEsconder; // lista a esconder    // Update is called once per frame  void Update () {  timeLeft -= Time.deltaTime; //reduzir o tempo  if (timeLeft < 0) // se o tempo restante for menor que zero  {  foreach (GameObject s in listaEsconder) { // para cada gameobject na lista a esconder    (s.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = -1; // mandar o sprite para a ordem -1 da layer  s.SetActive (false); // tornar objeto inativo  }  }  }  }  **4.6.2 – morte.cs**  using UnityEngine;  using System.Collections;  public class morte : MonoBehaviour {  // DECLARA E INICIALIZA VARIAVEIS  private Animator animator;    public GameObject[] exibirGUI;  public GameObject[] inimigo;  private bool morto;  public bool boss;  void Awake ()  {  animator = gameObject.GetComponentInChildren<Animator> ();  }  // FUNÇAO CHAMADA ATRAVES DE OUTRO SCRIPT, QUANDO ATIVA DEVE FAZER COM QUE OS INIMIGOS ESQUECAM DO JOGADOR  public void jogadorMorre()  {  if (morto == false) {  //Debug.Log("atingido");  morto = true;  //inimigo = GameObject as GameObject[];  foreach (GameObject inim in inimigo) {  if(inim == null) continue;  if (boss == false)  {inim.gameObject.GetComponent<AIWayPointsCS>().IgnorarPlayer();}  if (boss == true)  {inim.gameObject.GetComponent<BossAI>().IgnorarPlayer();}  }    }  // QUANDO SETADO COMO TRUE, OS COMPONENTES DO JOGADOR DEVEM SER DESTRUIDOS E TOCARA A ANIMACAO DE MORTE DO JOGADOR, BEM COMO O AVISO NA TELA  if (morto == true) {  foreach (GameObject tela in exibirGUI) {  tela.SetActive (true);  }    Component objAI = GetComponent ("movimentacao");  Component objAI2 = GetComponentInChildren<rotate> ();  Component objAI3 = GetComponent<CircleCollider2D> ();  Component objAI4 = GetComponentInChildren<Weapon> ();  Component objAI5 = GetComponentInChildren<ControleAnim> ();  Destroy (objAI);  Destroy (objAI2);  Destroy (objAI3);  Destroy (objAI4);  Destroy (objAI5);  animator.Play ("Marcos - Dying");  }    }    // AO COLIDIR COM O INIMIGO, JOGADOR SE TORNA "MORTO", SEUS COMPONENTES SAO DESTRUIDOS E A ANIMACAO DE MORTE SERA EXECUTADA, BEM COMO O AVISO NA TELA  void OnCollisionEnter2D(Collision2D outroColisor)  {  if(outroColisor.gameObject.tag == "Enemy") // se o colisor2D deste objeto colidir com um colisor com a tag Player  {  if (morto == false) {  foreach (GameObject inim in inimigo) {  inim.gameObject.GetComponent ("AIWayPointsCS").SendMessage ("IgnorarPlayer");  }  morto = true;  }  if (morto == true) {  foreach (GameObject tela in exibirGUI) {  tela.SetActive (true);  }    Component objAI = GetComponent("movimentacao");  Component objAI2 = GetComponentInChildren<rotate>();  Component objAI3 = GetComponent<CircleCollider2D>();  Component objAI4 = GetComponentInChildren<Weapon>();  Component objAI5 = GetComponentInChildren<ControleAnim>();  Destroy(objAI);  Destroy(objAI2);  Destroy(objAI3);  Destroy(objAI4);  Destroy(objAI5);  animator.Play ("Marcos - Dying");  }  }  }  }  **4.6.2 – movimentacao.cs**  using UnityEngine;  using System.Collections;  public class movimentacao : MonoBehaviour {    public float velocidade = 3; // velocidade de movimento    void FixedUpdate () {  rigidbody2D.velocity = new Vector2 (Input.GetAxis ("Horizontal") \* velocidade, Input.GetAxis ("Vertical") \* velocidade);  // move o rigidbody2D de acordo com a direcao e velocidade definidos  }  }  **4.6.2 – ProximosWaypointsCS.cs**  using UnityEngine;  using System.Collections;  public class ProximosWaypointsCS : MonoBehaviour {  // Criar uma lista de Objetos do TIPO GameObject!  // Uma lista e' marcada pelos []  public GameObject[] listaProximos;    public Transform ObterProximaPosicao () {  int indiceRandomicoDaListaObjetos = Random.Range(0,listaProximos.Length); // lista aleatoria de waypoints  GameObject objetoObtido = listaProximos[indiceRandomicoDaListaObjetos]; // inclui objeto na lista aleatoria  Transform transformObjeto = objetoObtido.transform; // objeto definido de forma aleatoria  return transformObjeto; // retonar qual objeto sera o proximo waypoint  }    }  **4.6.2 – rotate.cs**  using UnityEngine;  using System.Collections;  public class rotate : MonoBehaviour {    private Vector3 mousePosOld = Vector3.zero; // zera posicao atual do mouse    void Update () {  Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition); // obtem a posicao do mouse    if (mousePosOld.Equals (mousePos) == false) // So rotaciona quando a posiçao antiga do mouse for diferente  {  transform.rotation = Quaternion.LookRotation (Vector3.forward, mousePos - transform.position); // rotaciona objeto em direcao ao mousePos  mousePosOld = mousePos; // altera a posicao atual do mouse  }  if (Mathf.Abs (Input.GetAxis ("HorizontalRotation")) > 0.1 || Mathf.Abs (Input.GetAxis ("VerticalRotation")) > 0.1) // se o eixo horizontal e vertical forem maiores do que 0.1  {  transform.rotation = Quaternion.LookRotation (Vector3.forward,new Vector3 (Input.GetAxis ("HorizontalRotation"),  Input.GetAxis ("VerticalRotation"),  transform.position.z));  // rotaciona o objeto de acordo com o eixo  }  }  }  **4.6.2 – setActive.cs**  using UnityEngine;  using System.Collections;  public class setActive: MonoBehaviour {    public float timeLeft = 30.0f; // tempo travado no cronometro  public GameObject[] listaTelasEsconder; // lista de telas que serao ocultas    void Update()  {  timeLeft -= Time.deltaTime; // contagem regressiva a partir do tempo do cronometro  if(timeLeft < 0) // se o tempo for menor que zero  {  foreach (GameObject s in listaTelasEsconder) { // para cada objeto listado a ser escondido    (s.GetComponent<SpriteRenderer> () as SpriteRenderer).sortingOrder = -1; // mandar os sprites para a posicao -1 da layer  s.SetActive (false); // e desativa os objetos  }  }  }  }  **4.6.2 – showScore.cs**  using UnityEngine;  using System.Collections;  public class showScore : MonoBehaviour {  // DECLARA E INICIALIZA VARIAVEIS  public Weapon score;  public bool telaPont;  public bool telaMortes;  void Awake ()  {  score = GameObject.FindObjectOfType <Weapon> ();  }  // Update is called once per frame  // EXIBIR PONTUACAO NA TELA DE GAME OVER OU NA TELA INGAME  void Update () {  if (telaPont) {  guiText.text = "" + score.pontuacao;  }  else if (telaMortes){  guiText.text = "" + score.inimigosMortos;  }else  guiText.text = "Pontuação: " + score.pontuacao;    }  }  **4.6.2 – timer.cs**  using UnityEngine;  using System.Collections;  public class Timer : MonoBehaviour {  public string nomeDaCena = "nome da cena"; //cena a ser carregada  public float timeLeft = 60.0f; // tempo para carregar a cena    void Update()  {  timeLeft -= Time.deltaTime; //reduzir o tempo  if(timeLeft < 0) // se o tempo restante for menor que zero  {  Application.LoadLevel (nomeDaCena); // mudar a cena  }  }  }  **4.6.2 – Weapon.cs**  using UnityEngine;  using System.Collections;  public class Weapon : MonoBehaviour {  // DECLARA E INICIALIZA VARIAVEIS  public float fireRate = 0; // quando pode atirar  public LayerMask whatToHit; // layers que o raycast atingira  public int pistolAmmo = 11;  public float timerShooting = 0.3f;  public int pontuacao = 0;  public int inimigosMortos = 0;  private Animator animator;  private HashIDs hash;  public bool isShooting = false;  float timeToFire = 0; // delay entre disparos  Transform firePoint; // origem do raycast          void Start() {  pontuacao = PlayerPrefs.GetInt ("ChaveSalvarPontos", 0);  inimigosMortos = PlayerPrefs.GetInt ("ChaveInimigosMortos", 0);    }    void Awake () {  firePoint = transform.FindChild ("FirePoint"); // encontrar a origem do raycast na hierarquia do objeto que possui este script  animator = GetComponent<Animator> ();  hash = gameObject.GetComponent<HashIDs> ();    }  // DEFINIR MUNICAO, DEVE SER CHAMADO POR OUTRO SCRIPT    public void getWeapon()  {  pistolAmmo = 11;    }    // Update is called once per frame  void Update () {  if (fireRate == 0) { // se a taxa de tiro for igual a zero  if (Input.GetButtonDown ("Fire1")) { // e o botao de tiro for pressionado  Shoot (); // executar a funcao Shoot  }  } else {  if (Input.GetButtonDown ("Fire1") && Time.time > timeToFire) // caso contrario, onde o botao de tiro esteja pressionado e o tempo for maior que o tempo para disparo  {  timeToFire = Time.time + 1 / fireRate; // aumentar o tempo para disparar  Shoot(); // executar a funcao Shoot    }  if (pistolAmmo == 0)  {    }  }  // COOLDOWN PARA PODER ATIRAR NOVAMENTE  if (isShooting == true) {  timerShooting -= Time.deltaTime;  if (timerShooting <= 0) {  isShooting = false;  }    } else if (isShooting == false)  {  timerShooting = 0.5f;  }    }  void Shoot ()  {  if (isShooting == false && pistolAmmo > 0) { // se o botao de tiro estiver pressionado e isShooting ele nao estiver atirando  animator.SetTrigger (hash.shootTrigger);  audio.Play();  Vector2 mousePosition = new Vector2 (Camera.main.ScreenToWorldPoint (Input.mousePosition).x, Camera.main.ScreenToWorldPoint (Input.mousePosition).y); // identifica posicao do mouse (destino)  Vector2 firePointPosition = new Vector2 (firePoint.position.x, firePoint.position.y); // identifica a origem do disparo  RaycastHit2D hit = Physics2D.Raycast (firePointPosition, mousePosition - firePointPosition, 100, whatToHit); // cria raycast com as coordenadas acima  pistolAmmo = pistolAmmo -1;    if (hit.collider.gameObject.tag == "Enemy") // se o raycast atingir objeto com tag Enemy  {  EnemyScript enemyScript = hit.collider.gameObject.GetComponent<EnemyScript> (); // acessa o EnemyScript do objeto  enemyScript.MatarInimigo (); // ativa a funcao MatarInimigo  pontuacao = pontuacao + enemyScript.pontosDerrota; // soma a pontuacao  if (enemyScript.boss == false) // se nao for boss  inimigosMortos = inimigosMortos + 1; // aumenta o numero de inimigos  if (enemyScript.boss == true) // se for o boss  {  if (enemyScript.BossHealth == 0) // e o hp for igual a zero  inimigosMortos = inimigosMortos + 1; // acrescenta inimigos mortos +1  }  }  // Debug.Log ("muniçao da pistola = " + pistolAmmo); // exibe municao no LOG  isShooting = true; // e definir que o jogador estiver atirando      }    }  } |
|  |

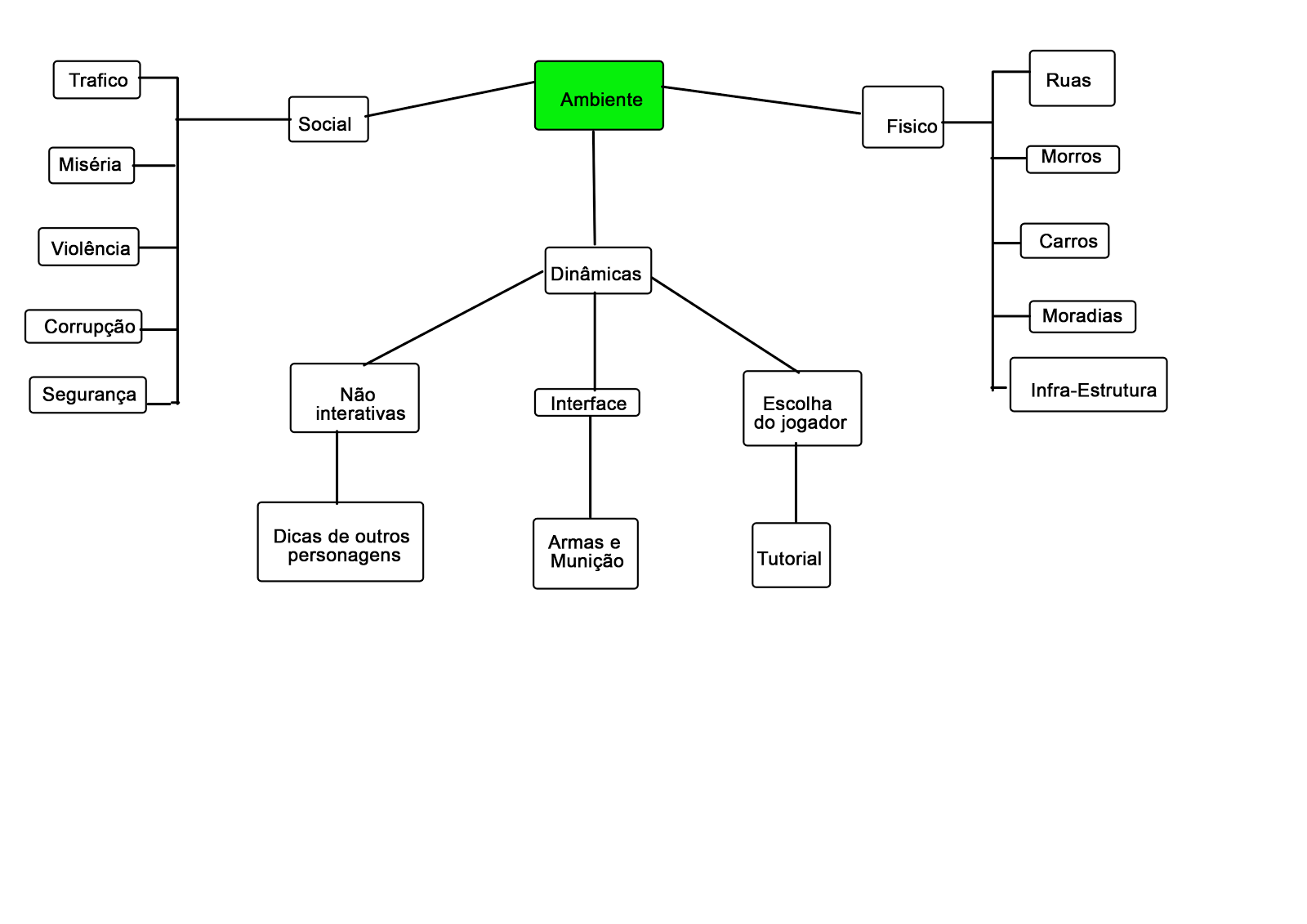
|  |
| --- |
| **5 – Softwares Secundários** |
| **5.1 – Editores**  Sons: Audacity e Audition CS6.  Programas de Arte: Photoshop CS5, Illustrator CS6, Corel DrawX5, Cpick.  Engine: Unity versão 4.5.2 |

|  |
| --- |
| **6 – Apêndices** |
| É recomendado escrever o caminho do arquivo com extensão para referências, inclusive.  **6.1 – Ativos de Arte**  **Lista de animações**  Em alguns momentos do jogo aparecerão animações de diálogo entre os personagens. - - Introdução ao jogo  - Entrada da casa de Matheus  - Encontro com o Boss  - Fim de Jogo  **Lista de efeitos**  - Fagulhas de de diferentes tamanhos (depende do porte da arma)  - Tiroteios  - Ambiente de festa possui troca de luzes, como em uma balada.  **Lista de interface artística**  - Arma, Munição e Pontuação  - Menu  **Lista de cenas de corte**  - Entrada da casa de Matheus - Encontro com o Boss - Fim de Jogo  **6.2 – Ativos de Som**  **Sons de ambiente**  Músicas instrumentais relacionados ao ambiente.  Sonoplastia de pássaros e vozes.  **Sons de armas**  Também de acordo com o porte da arma de fogo, acompanhado de sons quando você engatilha e recarrega. Há também o som do soco se movendo no vento.  **Sons de interface**  Não possui, somente uma musica tocando de background  **6.3 – Ativos de Música**  **Ambiente**  Musicas instrumentais relacionados ao ambiente.  **“Ação”**  Não.  **Vitória**  Não.  **Derrota**  Não.  **6.5 – Mapas conceituais**  **Personagem**  GDD.png |

**Jogabilidade**



**Ambiente**



|  |  |  |
| --- | --- | --- |
| |  | | --- | | Em nn / nn / nnnn | | **Nome do Gerente do Projeto**  Gerente do Projeto | |
| |  | | --- | | Em nn / nn / nnnn | | **Nome do Coordenador Técnico**  Coordenador Técnico | |
| |  | | --- | | Em n / nn / nnnn | | **Nome do Coordenador Artístico**  Coordenador Artístico | |