Curs 2017 - 2018

Querfer Quest

PROGRAMACIÓ D'UN VIDEOJOC AMB UNITY

GERARD QUERALT FERRÉ

INSTITUT SÒL-DE-RIU | Tutor: Claudio Cosín Sylla

Índex

Introducció

En el meu treball de recerca he programat un videojoc del gènere plataformes. L'he programat utilitzant el motor Unity, molt emprat en la indústria dels videojocs, i amb el llenguatge de programació C#, del qual m'havia proposat comprendre'n la base.
A banda de la programació, també he construït les pantalles i els personatges, tot a partir d'"assets" -dels quals en parlaré en profunditat més endavant-, i he composat personalment la música a partir de la d'altres jocs.

Tria del tema

He escollit dissenyar un videojoc per al meu treball de recerca per diversos motius, però el més important potser sigui que, volent-me dedicar a la programació de videojocs, puc utilitzar aquesta oportunitat per aprendre no només un llenguatge de programació àmpliament utilitzat com és Unity, sinó tot el conjunt del procés de creació d'un videojoc. Tot i centrar-me en la programació, també he pogut fer un tast de coses com ara disseny de mapes o composició de la música, que pot resultar-me útil si acabo en una posició de cap d'equip. Principalment, però, em resultava molt atractiva la idea d'aprendre a programar amb Unity, sent aquest un dels llenguatges més utilitzats, i estic segur que, per superficial que hagi estat l'experiència, m'ha resultat prou enriquidora.

A banda de tots els avantatges pràctics, també vaig creure important escollir un treball que em resultés divertit de fer, cosa que sumat a tot el que m'aportava a nivell curricular va fer que la idea del joc fos la més idònia.

A l'hora d'escollir el tipus de joc, en aquest cas un plataformes, ho vaig fer per motius purament pràctics. No només són els plataformes relativament senzills de programar, sinó que també fan us de les físiques, cosa que vam decidir amb el tutor que era una cosa amb la que volíem treballar; un plataformes em va semblar la tria lògica.

Objectius

Principals

- Familiaritzar-me amb Unity i el seu mètode de treball
- Assolir un nivell molt bàsic de C#
- Comprendre tots els processos creatius darrere d'un joc
- Fer un joc que a mi m'agradaria jugar

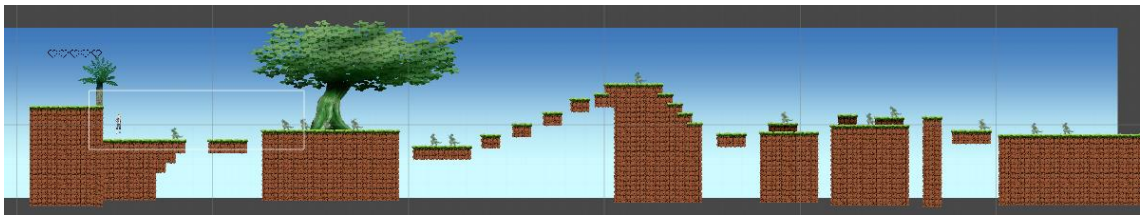
Secundaris

- Fer un joc el més atractiu visualment possible
- Fer una banda sonora respectable

Desenvolupament de la idea

Com ja he dit, vaig decidir que el joc seria un plataformes perquè em va semblar assequible a banda d'adequat per als objectius plantejats. Decidit això, vaig començar a plantejar-me com seria el joc com a tal. En primer lloc vaig escollir fer-lo en 2D, a banda de perquè es l'estil dels grans clàssics del gènere, també perquè em resultava més senzill a l'hora de fer les físiques, molt complexes d'altra manera. Clara la base, em vaig plantejar fer una història, donat que m'agrada escriure coses de tant en tant, però ho vaig descartar al cap d'uns pocs esborranys perquè no em veia capaç de fer una història més o menys seriosa sense que quedés molt fora de lloc, donada la senzillesa del joc com a tal.

Com que tampoc volia fer una cosa molt senzilla, vaig pensar d'afegir-hi diferents pantalles, les quals volia que tinguessin temàtica diferent però amb certa cohesió. Vaig decidir fer els clàssics escenaris d'una jungla, un volcà i una glacera, i un castell com a escenari final.



Pla general del nivell de la jungla

Per a fer tant el jugador com els monstres, vaig buscar a la tenda de Unity, la qual ofereix els anomenats "assets" fets per la comunitat. Els "assets" són uns paquets que fan els usuaris de Unity i posen a disposició de la resta d'usuaris, ja sigui de manera gratuïta o amb una compensació econòmica. Com que no disposo de ningú que em pugui fer els dissenys, vaig haver de recórrer a aquest mètode, i vaig obtenir millors resultats dels que esperava. En el meu cas vaig tenir la sort de trobar "assets" gratuïts tant per al jugador com per als monstres fets pel mateix usuari (acreditat en l'annex), aportant una consistència visual que no creia que podria assolir i que crec que és molt d'agrair.

Trobats els personatges, vaig buscar uns "assets" per a cadascuna de les pantalles. Tot i adonar-me llavors que hauria d'haver pensat les pantalles un cop en tingués els "assets", en no ser idees gaire originals, no em va costar

trobar-los per a les quatre pantalles que tenia en ment, fets també les quatre pel mateix usuari.

En quant a la música, aquesta sí que va ser de collita pròpia, aprofitant la meva anterior experiència component algunes cançons i l'inestimable ajuda del mestre Juanjo Villarroya, que ha compost música per a empreses tant importants com King, i a qui tinc el plaer de tenir com a professor, vam composar una cançó per a cada pantalla, així com una cançó per al menú principal, tot basant-nos en cançons d'alguns dels meus jocs preferits.

Per últim, però no menys important, calia posar-li nom al joc. Com que trobar noms originals no és el meu punt fort, vaig decidir inspirar-me en una de les meves sèries preferides, "New Game!!", on una noia, Nene Sakura, programa ella sola un joc, i li posa de nom "Nene Quest". En part també com a tribut per la motivació que em donava quan no tenia ganes de seguir veient tutorials, vaig decidir posar al joc "Querfer Quest", sent Querfer el "nick" que utilitzo on-line (combinació de Queralt i Ferré).

Esbossos de dues versions del jugador amb diferents expressions

Eines utilitzades

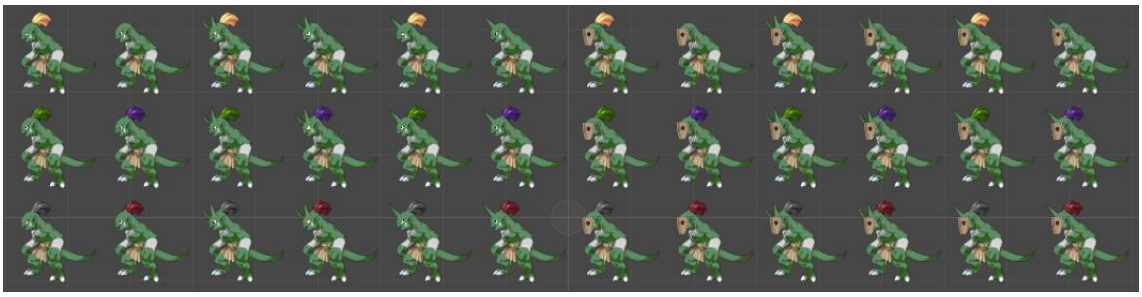
El joc està construït amb Unity, un programa especialitzat en la creació de videojocs per a ordinador, consola i telèfon mòbil. Unity va ser publicat el 2005 per Unity Technologies exclusivament per a macOS, i des de llavors s'ha estès per 27 plataformes diferents. Per al codi he treballat amb el llenguatge C#, fet per Microsoft i d'ús estàndard en Unity.

Els tutorials que he visualitzat per a aprendre dit llenguatge els he trobat a través de la mateixa web de Unity o per YouTube, on estan penjats també els oficials.

Per a la composició de la música he treballat amb el programa Finale, àmpliament utilitzat per a fer cançons. Publicat originalment el 1988 per MakeMusic, permet escriure i executar partitures de qualsevol tipus, des de bandes i orquestres fins a peces individuals, i amb el qual ja havia treballat anteriorment a l'Escola Municipal de Música d'Alcanar.

Procés

Un cop decidit a fer un plataformes, vaig començar amb els tutorials més bàsics de Unity, on vaig fer un joc molt senzill que em va ajudar a familiaritzar-me amb coses com el moviment del personatge, la detecció de col·lisions o el funcionament de la càmera. Acabats aquests, vaig començar els tutorials específics per als plataformes en 2D. A banda, vaig buscar “assets” a la tenda de Unity per a fer tant els personatges com les pantalles.
A l’hora de fer els dissenys de personatges, vaig provar amb dues versions del protagonista, una més “normal”, per dir-ho d’alguna manera, i una a poder més estrafolària, pròpia de molts jocs actuals; finalment, però, em vaig decidir per la idea inicial. En els monstres vaig veure l’oportunitat de donar més vida al joc; donades les limitacions artístiques que tenia, havent de dependre d’”assets”, vaig aprofitar els monstres per a donar al joc una sensació de profunditat.



Disseny dels monstres de la jungla

Concretament, aprofitant les moltes possibilitats que em brindava l’”asset”, vaig dissenyar enemics el més diversos possibles, especialment per la pantalla de la jungla, a banda de perquè pensava reciclar-los al llarg del joc, perquè creia que per a donar una sensació de tribu de la jungla s’havien de veure molt caòtics i diversos.

Disseny dels monstres del volcà

Disseny dels monstres del gel

Per als altres dissenys vaig provar de fer-los de manera lògica: per al volcà, en ser més civilitzats, vaig reduir la varietat per donar una sensació d'unitat, de que tots pertanyen al mateix grup; vaig fer el mateix per als monstres del món de gel, amb la diferència que d'aquests no en vaig fer cap sense pèl perquè em semblava absurd que ningú que visqués en una glacera decidís tallar-se'l; a aquests també els hi vaig afegir uns ulls més intel·ligents per a que donessin la sensació de ser més perillosos que els monstres anteriors tot i no semblar diferents físicament.

Disseny del guàrdia del castell

Per últim, dels guàrdies del castell final en vaig fer una única versió per accentuar aquesta uniformitat, pròpia dels cavallers medievals, a banda del fet que pretenia reutilitzar els models anteriors i no volia abrumar el jugador amb tot de monstres cadascun diferent als altres.

Per als "bosses", vaig provar d'imaginar-me com seria el cap de cada tribu, i vaig fer com versions millorades de cadascun dels monstres afegint-los alguns detalls per a donar-los més personalitat, com els ulls sense pupil·les del "boss" de la jungla, que donen una sensació de "berserker" assedegat de sang. Per al "boss" final vaig provar de fer una versió "fosca" del protagonista, que pensava justificar amb algun tipus d'història que, com ja he comentat, vaig decidir descartar. La idea d'aquesta contrapart fosca del protagonista està inspirada sobretot en el personatge de Dark Pit, literalment el reflex de la part fosca de Pit, el protagonista del joc Kid Icarus: Uprising, un dels meus preferits.

Disseny dels "bosses", d'esquerra a dreta: jungla, volcà, gel i castell

Creats els personatges, vaig començar a banda a composar la música per al joc, amb la ja esmentada ajuda del mestre Villarroya. Vaig presentar-li algunes de les meves cançons preferides dels jocs que més he disfrutrat, i vam agafar-ne la base per a fer una cançó original. Per al tema del menú principal tenia clar que volia basar-me en la cançó “Pilgrims on a long journey”, del joc Child of Light; per a la jungla ens vam basar en el famós “Jungle Hijinx”, de Donkey Kong; per al volcà vam adaptar el tema del Volcà d’Eldin de l’espectacular banda sonora de The Legend of Zelda: Skyward Sword; per a la glacera vaig proposar els temes de dues zones de l’expansió de World of Warcraft: Wrath of the Lich King, Dragonblight i Icecrown; finalment ens vam decantar per aquest últim, molt més marcial. Per a la zona final volia dos cançons diferents, una per al castell com a tal i l’altra per al combat contra el “boss” final, i per a seguir el patró de joc per zona, li vaig proposar diverses cançons de l’increïble banda sonora del ja esmentat Kid Icarus: Uprising: per al castell, els temes “Underworld Castle”, “Hades’ Infernal Theme” i “Dark Pit”, i per al combat final, “Medusa’s Final Battle” i altre cop “Hades’ Infernal Theme” i “Dark Pit”, que també em semblaven apropiats per al combat. Finalment ens vam decidir per “Dark Pit” per al tema final, i ens vam basar en la cançó que havíem fet a l’hora de composar la del castell.

Disseny del "boss" final

Per a fer les plataformes de cada pantalla vaig procurar que anés augmentant la dificultat a mesura que s’avançava, així com donar vida a l’escenari amb l’ús de diversos fons.

Conclusions

El treball ha sigut sens dubte molt més difícil i tediós del que m'esperava, però un cop vaig agafar el ritme es va fer més lleuger. Tot i que hi ha coses que he après en els tutorials que no he acabat d'entendre per una manca de base, he entès la majoria dels conceptes. A més m'he familiaritzat bastant amb la interfície de Unity, que era un dels objectius que m'havia proposat. Dissenyar els personatges va ser molt divertit i distès, malgrat no haver comptat amb dissenys propis. Igualment, fer les pantalles i provar cada vegada salts més difícils (no només planejar-los sinó comprovar que fossin possibles) va ser molt entretingut de fer, com també va ser la música tot i anar una mica justos per acabar-la.

Així, la part més dura va ser la de programació, la base del treball. Va ser difícil comprendre alguns conceptes més complexos, cosa que volia provar de fer per a no tenir la sensació que gastava el temps per a res. Val a dir que el fet d'haver de aprendre a través de vídeos va ser la que va suposar bastants problemes, perquè molts dels tutorials partien d'una base que jo no tinc. Al final, però, crec que me n'he sortit prou bé, i estic molt content del resultat. Tot i distar molt de ser un joc professional, crec que és un digne primer intent.

Annex

Scripts

Motor de Físiques

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PhysicsObject : MonoBehaviour {

    public float minGroundNormalY = .65f;
    public float gravityModifier = 1f;

    protected Vector2 targetVelocity;
    protected bool grounded;
    protected Vector2 groundNormal;
    protected Rigidbody2D rb2d;
    protected Vector2 velocity;
    protected ContactFilter2D contactFilter;
    protected RaycastHit2D[] hitBuffer = new RaycastHit2D[16];
    protected List<RaycastHit2D> hitBufferList = new List<Raycas
tHit2D> (16);

    protected const float MinMoveDistance = 0.001f;
    protected const    float shellRadius = 0.01f;

    void OnEnable()
    {
        rb2d = GetComponent<Rigidbody2D> ();
    }

    void Start ()
    {
        contactFilter.useTriggers = false;
        contactFilter.SetLayerMask (Physics2D.GetLayerCollisionM
ask (gameObject.layer));
        contactFilter.useLayerMask = true;
    }

    //Cada “frame”, tornarem a calcular la velocitat
    void Update ()
    {
        targetVelocity = Vector2.zero;
        ComputeVelocity ();
    }

    protected virtual void ComputeVelocity ()
    {

    }
```

```
    //Calculem la velocitat a través dels paràmetres establers
als començament del codi, i també si està o no al terra (que
utilitzarem per al control del jugador)
    void FixedUpdate ()
    {
        velocity += gravityModifier * Physics2D.gravity * Time.d
eltaTime;
        velocity.x = targetVelocity.x;

        grounded = false;

        Vector2 deltaPosition = velocity * Time.deltaTime;

        Vector2 moveAlongGround = new Vector2 (groundNormal.y, -
groundNormal.x);

        Vector2 move = moveAlongGround * deltaPosition.x;

        Movement (move, false);

        move = Vector2.up * deltaPosition.y;

        Movement (move, true);
    }

    void Movement (Vector2 move, bool yMovement)
    {
        float distance = move.magnitude;

        if (distance > MinMoveDistance)
        {
            int count = rb2d.Cast (move, contactFilter, hitBuffe
r, distance + shellRadius);
            hitBufferList.Clear ();
            for (int i = 0; i < count; i++){
                hitBufferList.Add (hitBuffer [i]);
            }

            for ( int i=0; i<hitBufferList.Count; i++)
            {
                Vector2 currentNormal = hitBufferList [i].normal
;
                if (currentNormal.y > minGroundNormalY)
                {
                    grounded = true;
                    if (yMovement)
                    {
                        groundNormal = currentNormal;
                        currentNormal.x = 0;
```

```
                    }
                }
                float projection = Vector2.Dot (velocity, curren
tNormal);
                if (projection < 0)
                {
                    velocity = velocity - projection * currentNo
rmal;
                }
                float modifiedDistance = hitBufferList [i].dista
nce - shellRadius;
                distance = modifiedDistance < distance ? modifie
dDistance : distance;
            }

        }

        rb2d.position = rb2d.position + move.normalized * distan
ce;
    }

}
```

Control Jugador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Utilitzarem funcions i paràmetres del motor de físiques, per
això creem el controlador dins d'aquest
public class PlayerPlatformerController : PhysicsObject {

    //A la consola determinarem la velocitat màxima a la que pot
anar el jugador, així com la velocitat que té en saltar
    public float maxSpeed;
    public float jumpTakeOffSpeed;

    public float knockback;
    public float knockbackLenght;
    public float knockbackCount;
    public bool knockFromRight;

    //A la consola donarem un valor a la vida màxima del
jugador, i més endavant en el codi igualarem la vida actual a
aquesta en iniciar el nivell
    public int currentHealth;
    public int maxHealth;

    void Awake ()
    {
        currentHealth = maxHealth;
    }
    //Creem una funció que reinicia el nivell per a quan mori el
jugador
    void Die()
    {
        Application.LoadLevel (Application.loadedLevel);
    }
    //Creem una funció que calcula la velocitat del jugador a
partir d'uns paràmetres i de l'"input" que rep del teclat
    protected override void ComputeVelocity()
    {
        Vector2 move = Vector2.zero;

        move.x = Input.GetAxis ("Horizontal");

        if (Input.GetButtonDown ("Jump") && grounded) {
            velocity.y = jumpTakeOffSpeed;
        } else if (Input.GetButtonUp ("Jump"))
        {
            if (velocity.y > 0)
                velocity.y = velocity.y * .5f;
```

```
        }
        //Només permetrem el moviment si el jugador no ha estat
empentat per un enemic, i calcularem aquí la distància on arriba
despès de l'empenta
        if (knockbackCount <= 0) {
            targetVelocity = move * maxSpeed;
        } else {
            if (knockFromRight)
                targetVelocity = new Vector2 (-
knockback, knockback);
            if (!knockFromRight)
                targetVelocity = new Vector2 (knockback, knockba
ck);
            knockbackCount -= Time.deltaTime;
        }
        {
            //Si la vida actual supera la màxima permesa,
igualem l'actual a la màxima
            if (currentHealth > maxHealth) {
                currentHealth = maxHealth;
            }
            //Si la vida actual és igual o inferior a zero
"cridem" la funció que reinicia el nivell
            if (currentHealth <= 0) {
                Die ();
            }
        }
    }

    //Quan el jugador és ferit li restem punts de vida iguals
als que determinarem més endavant (Apartat "Ferir Jugador"), i
activem una animació que el fa tornar vermell uns segons
    public void Damage(int dmg)
    {
        currentHealth -= dmg;
        gameObject.GetComponent<Animation> ().Play ("Player_RedF
lash");
    }

}
```

Vida Jugador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class HUD : MonoBehaviour {

    //Creem una matriu per a emmagatzemar les diferents imatges dels cors, que canviaran a mesura que baixi la vida
    public Sprite [] HeartSprites;

    //Indiquem a la consola on estan els cors que s'han de canviar
    public Image HeartUI;

    //Creem una variable que "cridarà" el codi per controlar el jugador, on hi ha els valors de la vida actual
    private PlayerPlatformerController player;

    void Start ()
    {
        player = GameObject.FindGameObjectWithTag ("Player").GetComponent<PlayerPlatformerController> ();
    }

    void Update ()
    {
        //A la consola numerem les "sprites" dels cors en funció del nombre de cors que hi ha, i així agafarà el nombre de la matriu pertinent; per exemple, si el jugador té 3 punts de vida, com que la imatge número 3 de la matriu són els tres cors el programa mostrarà la salut correctament
        HeartUI.sprite = HeartSprites[player.currentHealth];
    }
}
```

Moviment Enemics

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyAI : MonoBehaviour {

    //A la consola donem un valor de velocitat i creem una
variable per saber si s'està movent cap a la dreta o no
    public float moveSpeed;
    public bool moveRight;

    //A la consola determinem un objecte que revisi les
col·lisions i qui objecte està considerat una paret, a banda de
crear una variable que utilitzarem més endavant
    public Transform wallCheck;
    public float wallCheckRadius;
    public LayerMask whatIsWall;
    private bool hittingWall;

    //Cada "frame" comprovem si l'objecte compleix unes
condicions que determinen que està en contacte amb una paret
    void Update () {

        hittingWall = Physics2D.OverlapCircle (wallCheck.positio
n, wallCheckRadius, whatIsWall);

        //Si s'està movent cap a la dreta en tocar una paret
canviarà de direcció i a l'inrevés
        if (hittingWall)
            moveRight = !moveRight;

        //Determinem a quina velocitat es mou
        if (moveRight) {
            GetComponent<Rigidbody2D> ().velocity = new Vector2
(moveSpeed, GetComponent<Rigidbody2D> ().velocity.y);
        } else
        {
            GetComponent<Rigidbody2D> ().velocity = new Vector2
(-moveSpeed, GetComponent<Rigidbody2D> ().velocity.y);
        }
    }

}
```

Vida Enemics

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyHealthManager : MonoBehaviour {

    //A la consola donarem un valor als punts de vida dels
enemics
    public int enemyHealth;

    //Quan la salut dels enemics sigui igual o inferior a zero
els eliminarem de l'escena
    void Update () {
        if (enemyHealth <= 0)
        {
            Destroy (gameObject);
        }
    }
    //Creem una funció que resta punts de vida als enemics en
funció dels punts que infligeix el jugador, i que utilitzarem en
el codi "Ferir Enemics"
    public void giveDamage(int damageToDeal)
    {
        enemyHealth -= damageToDeal;
    }
}
```

Ferir Jugador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyDamage : MonoBehaviour {

    //"Cridem" el codi del jugador per a utilitzar-lo en aquest
    private PlayerPlatformerController player;

    //A la consola introduïm el valor de punts de vida que
traurem al jugador
    public int DmgDealt;

    void Start ()
    {
        player = GameObject.FindGameObjectWithTag ("Player").Get
Component<PlayerPlatformerController>();
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        //Si aquest objecte està marcat com a jugador, farem
dues coses: en primer lloc li restarem punts de vida iguals als
que hem determinat anteriorment, i li donarem una empenta
        if(col.CompareTag("Player"))
            {
            player.Damage(DmgDealt);

            player.knockbackCount = player.knockbackLenght;
            if (player.transform.position.x < transform.position
.x)
                player.knockFromRight = true;
            else
                player.knockFromRight = false;
            }
    }
}
```

Ferir Enemics

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HurtEnemy : MonoBehaviour {

    //A la consola introduïm el valor de punts de vida que
traurem a l'enemic en esclafar-lo
    public int damageToDeal;

    //Detectem si el jugador entra en contacte amb algun objecte
    void OnTriggerEnter2D(Collider2D other)
    {
        //Si aquest objecte està marcat com a enemic, restarem
als seus punts de vida la quantitat determinada anteriorment
        if (other.tag == "Enemy")
            other.GetComponent<EnemyHealthManager> ().giveDamage
 (damageToDeal);
    }
}
```

Càmera

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraFollow : MonoBehaviour {
    //Donarem uns valors que permetran que la càmera tingui un
moviment més suau
    private Vector2 velocity;
    public float smoothTimeY;
    public float smoothTimeX;
    //Vincularem la càmera al jugador per a que el segueixi
    public GameObject player;

    //A la consola determinarem si volem o no límits on la
càmera no podrà passar
    public bool bounds;
    //I els hi donarem unes coordenades
    public Vector3 minCameraPos;
    public Vector3 maxCameraPos;

    void Start () {
        player = GameObject.FindGameObjectWithTag ("Player");

    }
    //Calculem la posició de la càmera, que seguirà al jugador
    void FixedUpdate()
    {
        float posX = Mathf.SmoothDamp (transform.position.x, pla
yer.transform.position.x, ref velocity.x, smoothTimeX);
        float posY = Mathf.SmoothDamp (transform.position.y, pla
yer.transform.position.y, ref velocity.y, smoothTimeY);
        transform.position = new Vector3 (posX, posY, transform.
position.z);
        //Si hi ha límits no deixarem que la càmera superi els
valors determinats
        if (bounds)
        {
            transform.position = new Vector3 (Mathf.Clamp (trans
form.position.x, minCameraPos.x, maxCameraPos.x),
                Mathf.Clamp (transform.position.y, minCameraPos.
y, maxCameraPos.y),
                Mathf.Clamp (transform.position.z, minCameraPos.
z, maxCameraPos.z));
        }
    }

}
```

Botons Menú

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ButtonManager : MonoBehaviour {

    //Des de la consola de Unity assignarem l'escena que
s'ha de carregar
    public void NewGameBtn(string newGameLevel)
    {
        SceneManager.LoadScene (newGameLevel);
    }

    //Tanquem l'aplicació
    public void ExitGameBtn()
    {
        Application.Quit ();
    }
}
```

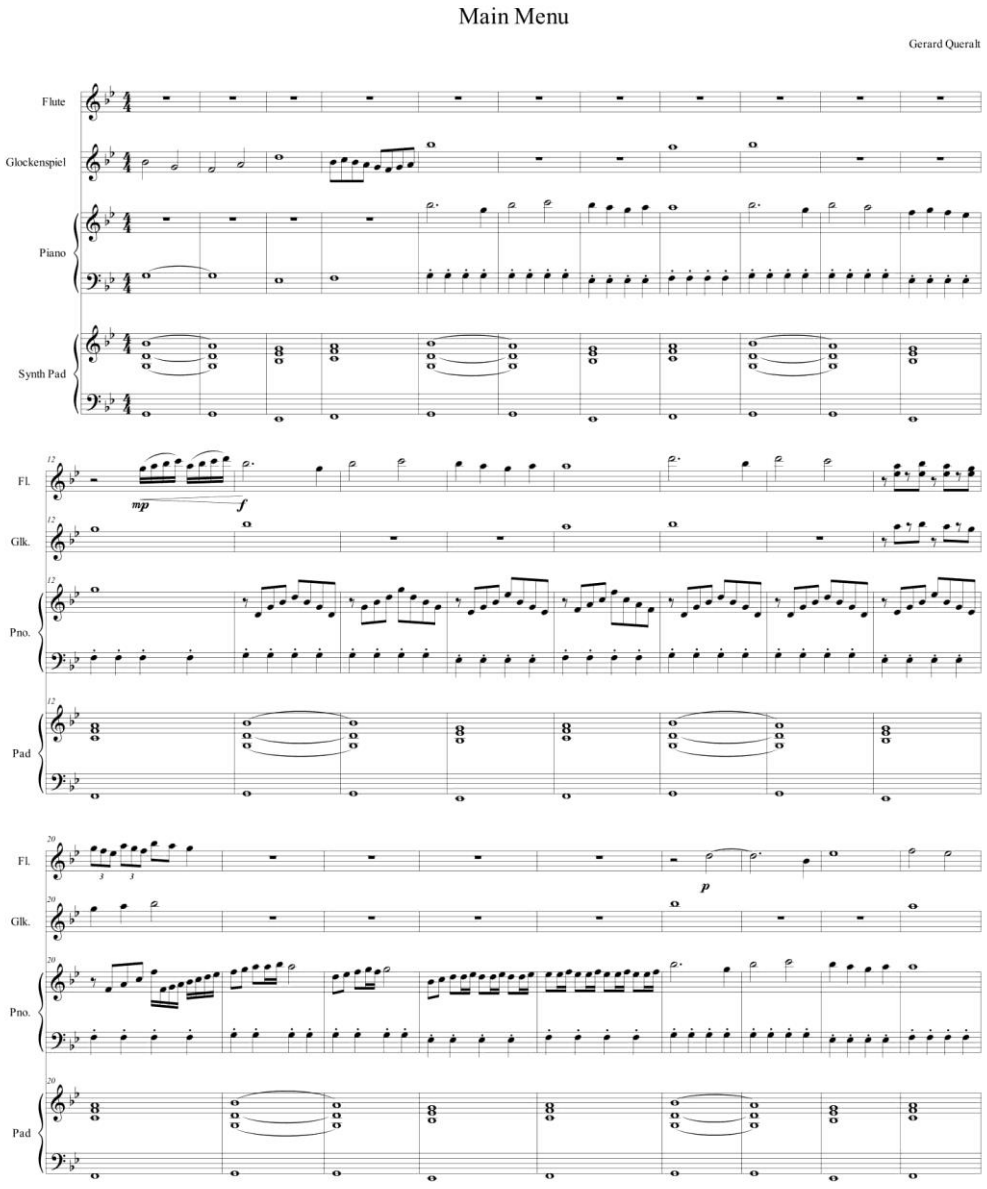
Menú Pausa

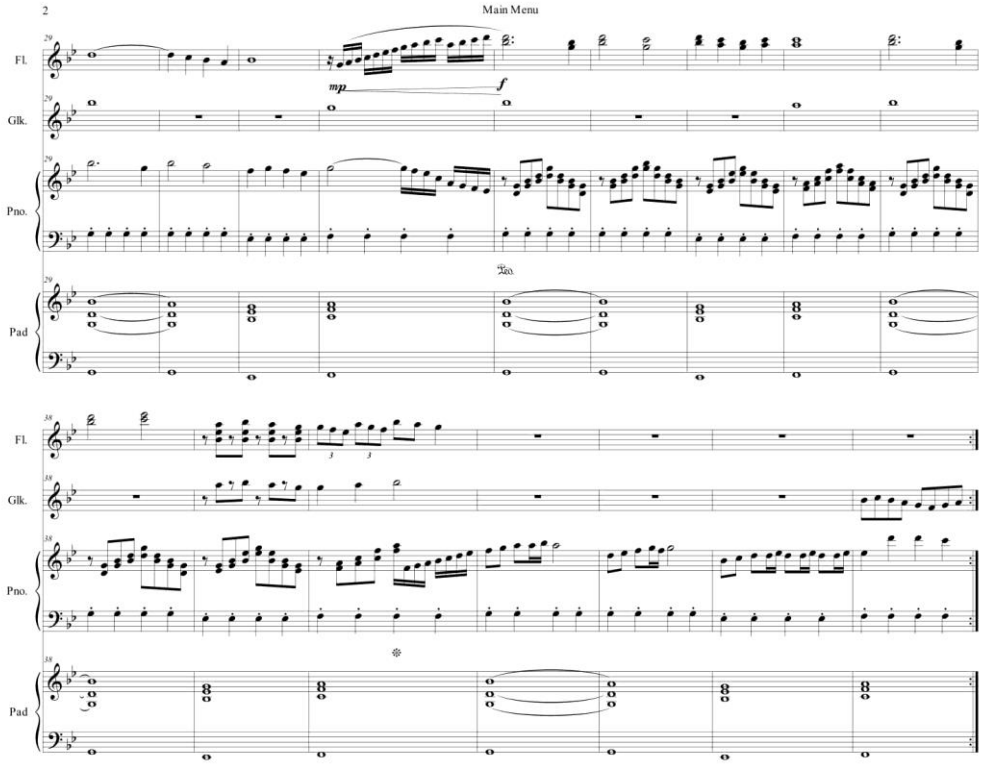
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PauseMenu : MonoBehaviour {

    public GameObject PauseUI;

    //De sèrie la pausa està desactivada
    private bool paused = false;

    void Start()
    {
        PauseUI.SetActive (false);
    }

    void Update()
    {
        //Quan es prem el botó de pausa se n'inverteix el valor
        if (Input.GetButtonDown ("Pause"))
        {
            paused = !paused;
        }

        //Si està pausat, activem la interfície de pausa i parem
el temps
        if (paused)
        {
            PauseUI.SetActive (true);
            Time.timeScale = 0;
        }
        //Si no està pausat, desactivem la interfície i tornem a
activar el temps
        if (!paused)
        {
            PauseUI.SetActive (false);
            Time.timeScale = 1;
        }

    }
    //Quan prenem aquest botó desactivem la pausa
    public void Resume ()
    {
        paused = false;
    }
    //Quan prenem aquest botó reiniciem el nivell actual
    public void Restart()
    {
```

```
        Application.LoadLevel (Application.loadedLevel);
    }
    //Quan prenem aquest botó anirem a l'escena 0, que en aquest
cas és el menú principal
    public void MainMenu()
    {
        Application.LoadLevel (0);
    }
    //Quan prenem aquest botó tanquem la aplicació
    public void Quit()
    {
        Application.Quit ();
    }

}
```

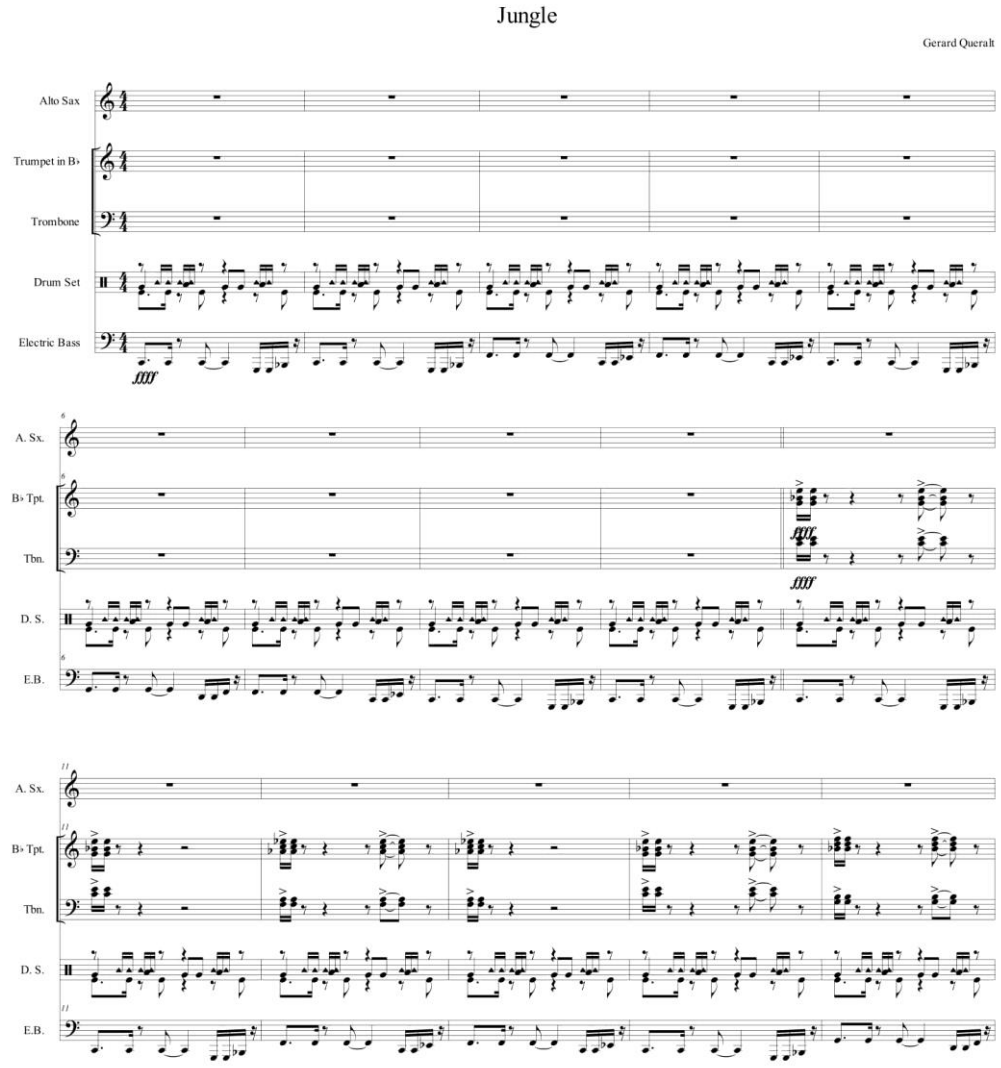
Partitures

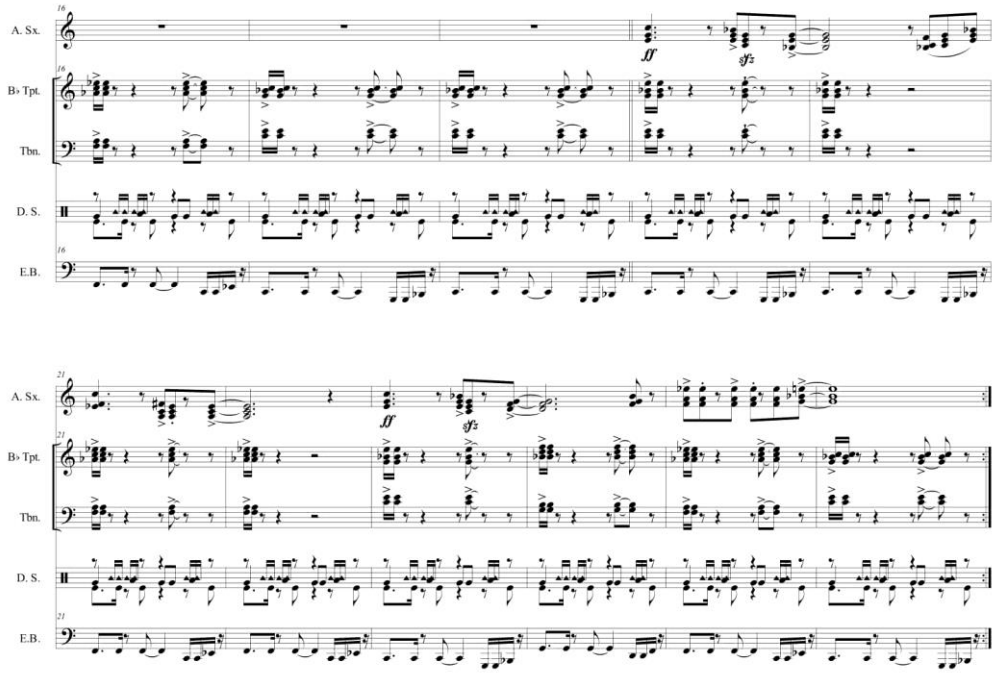
Menú principal




2
Main Menu
Fl.
Glk.
Pno.
Pad
mp
f
Ped.
Fl.
Glk.
Pno.
Pad

Jungla

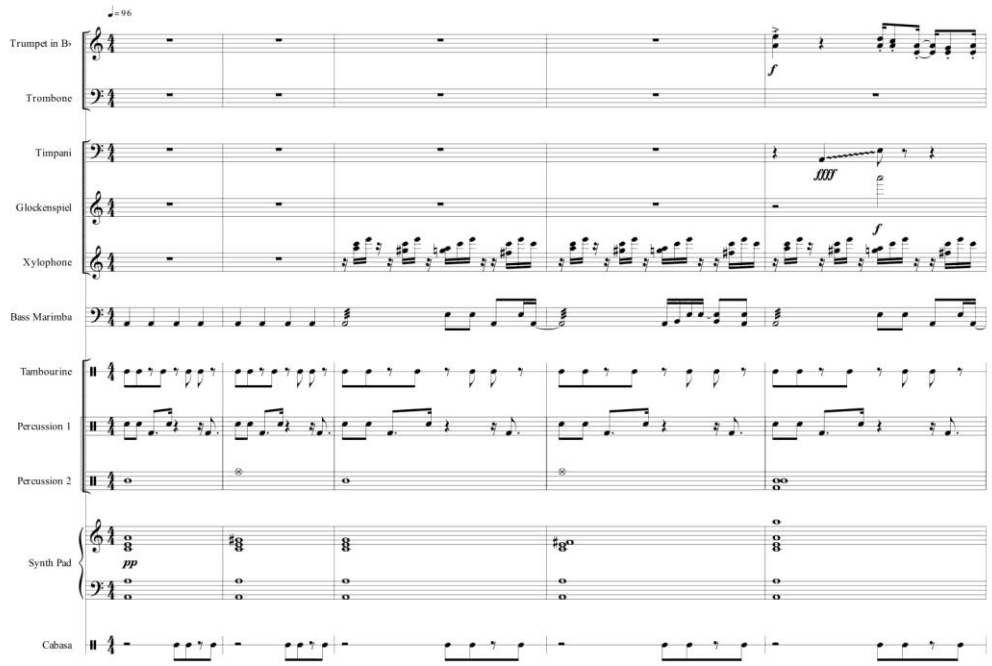

Jungle
Gerard Queralt
Alto Sax
Trumpet in B♭
Trombone
Drum Set
Electric Bass
A. Sx.
B♭ Tpt.
Tbn.
D. S.
E.B.

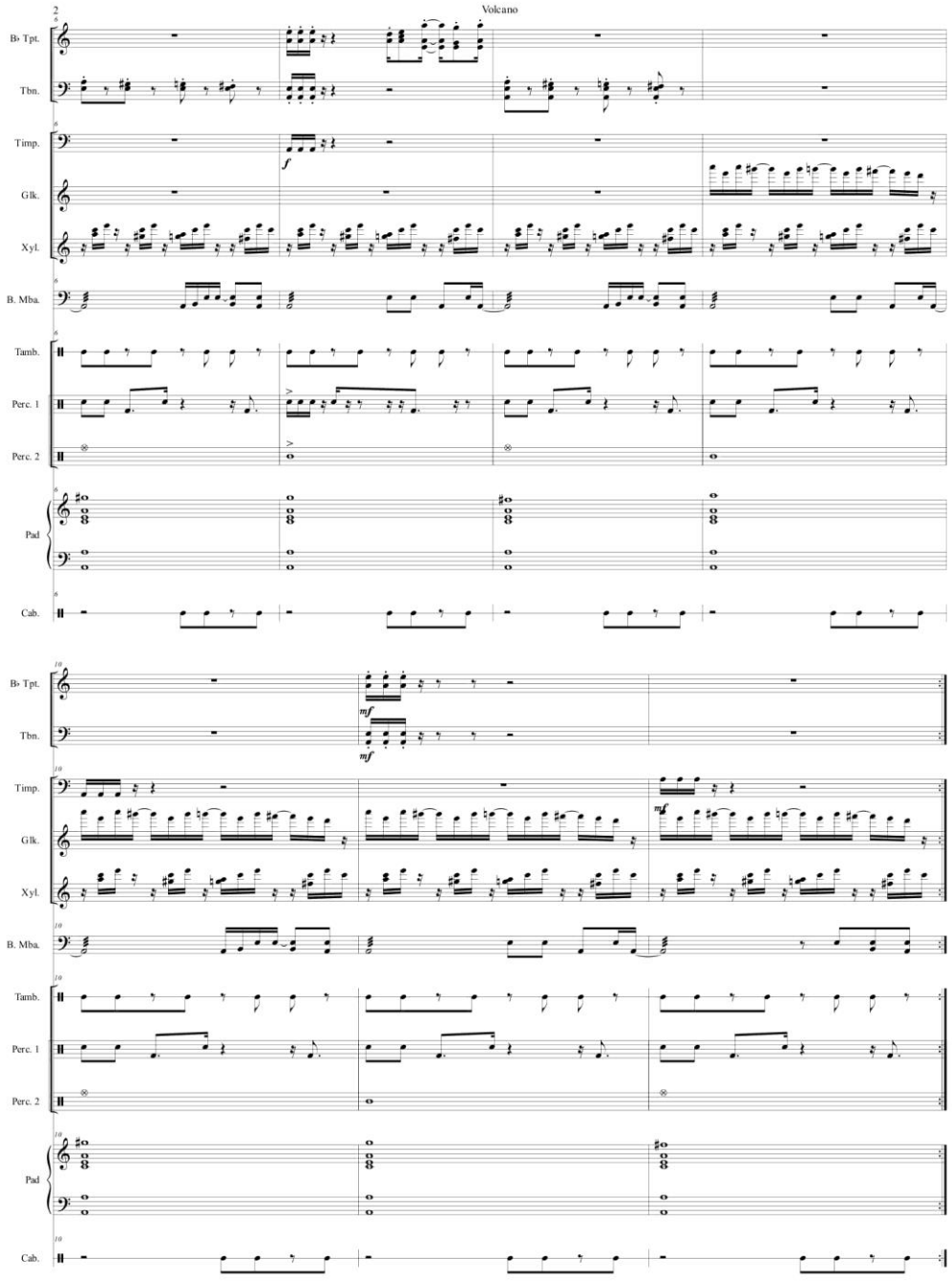

16
A. Sx.
B♭ Tpt.
Tbn.
D. S.
E.B.
ff
21
A. Sx.
B♭ Tpt.
Tbn.
D. S.
E.B.
ff

Volcà

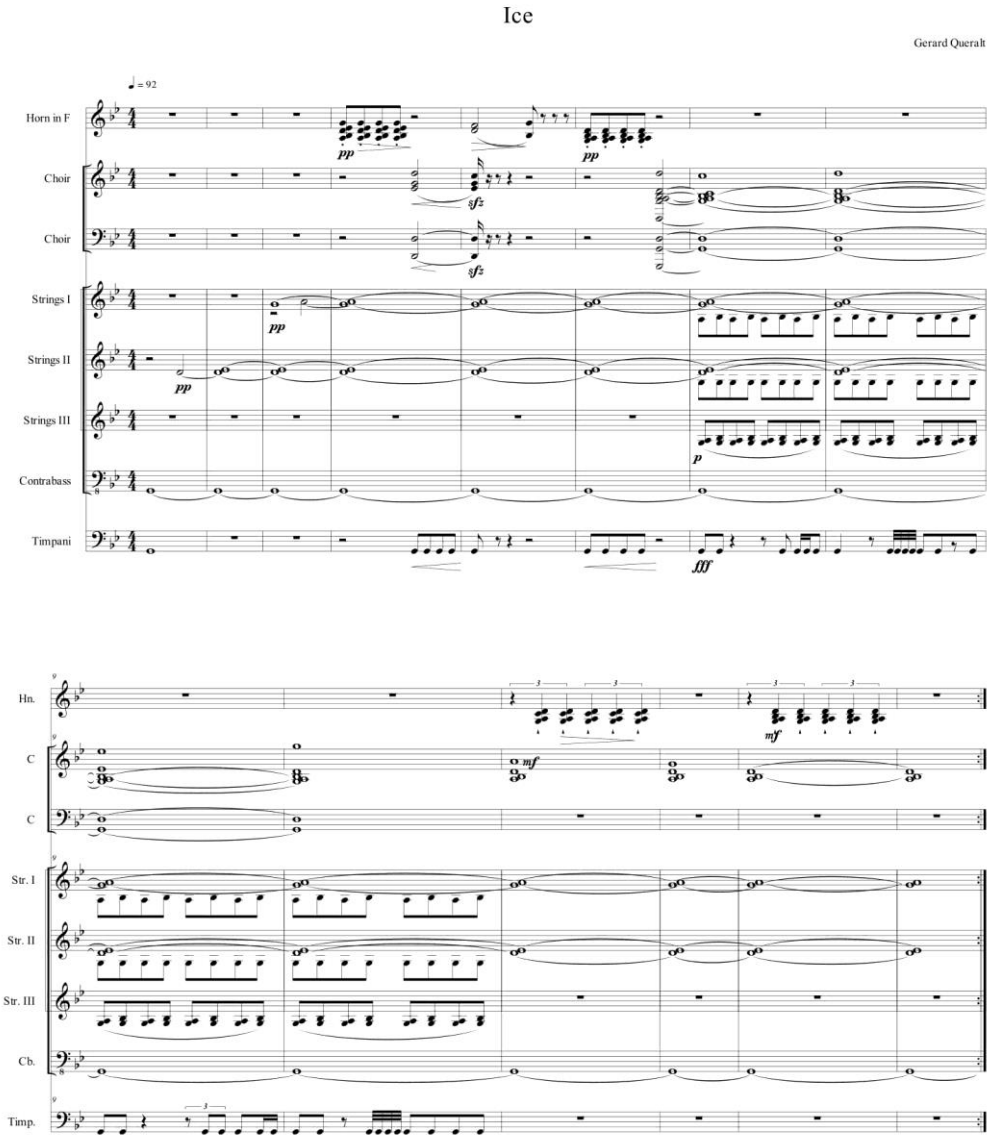
Volcano

Gerard Queralt

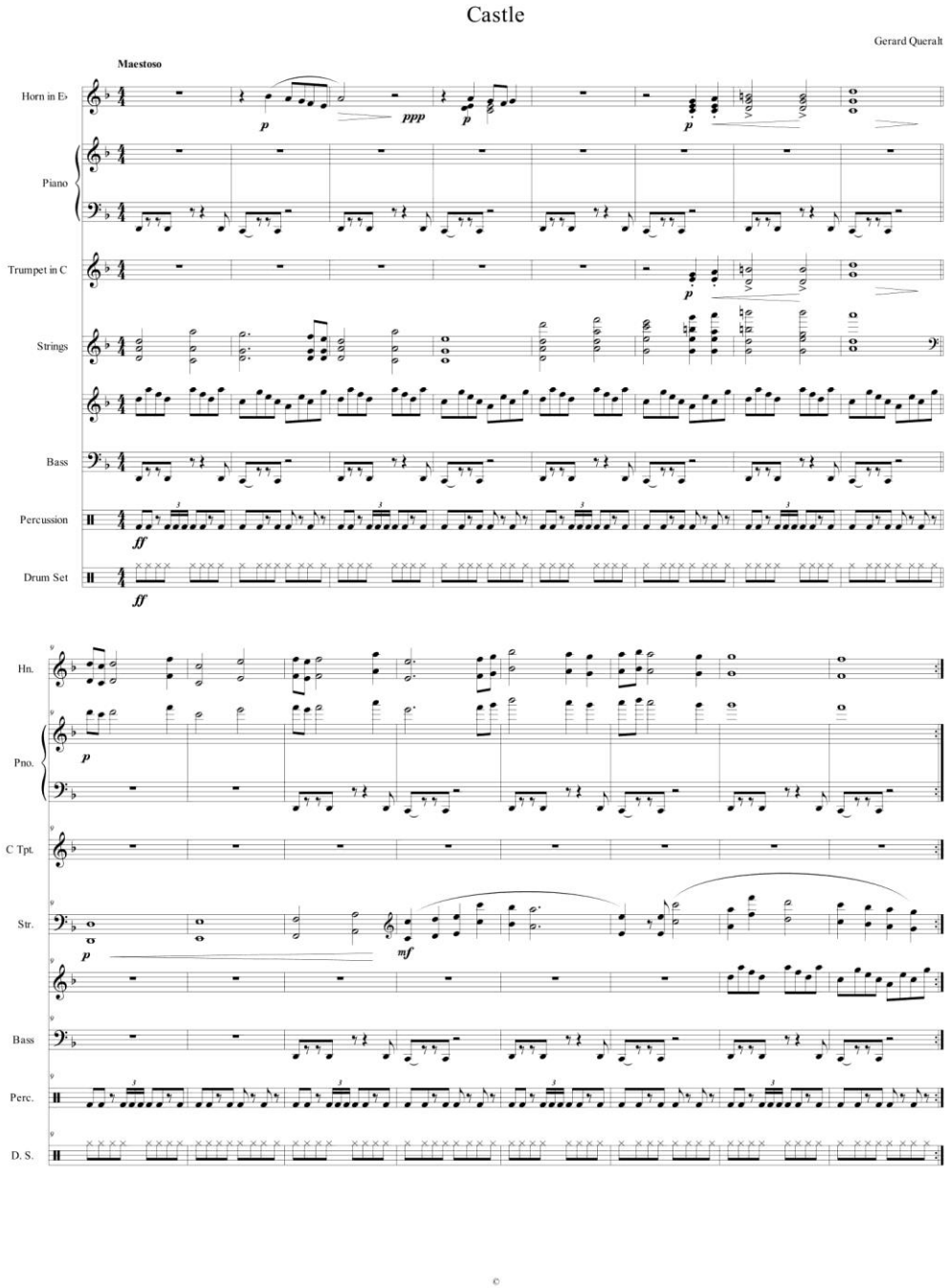



Volcano
B♭ Tpt.
Tbn.
Timp.
Glk.
Xyl.
B. Mba.
Tamb.
Perc. 1
Perc. 2
Pad
Cab.
f
mf
mf
mf

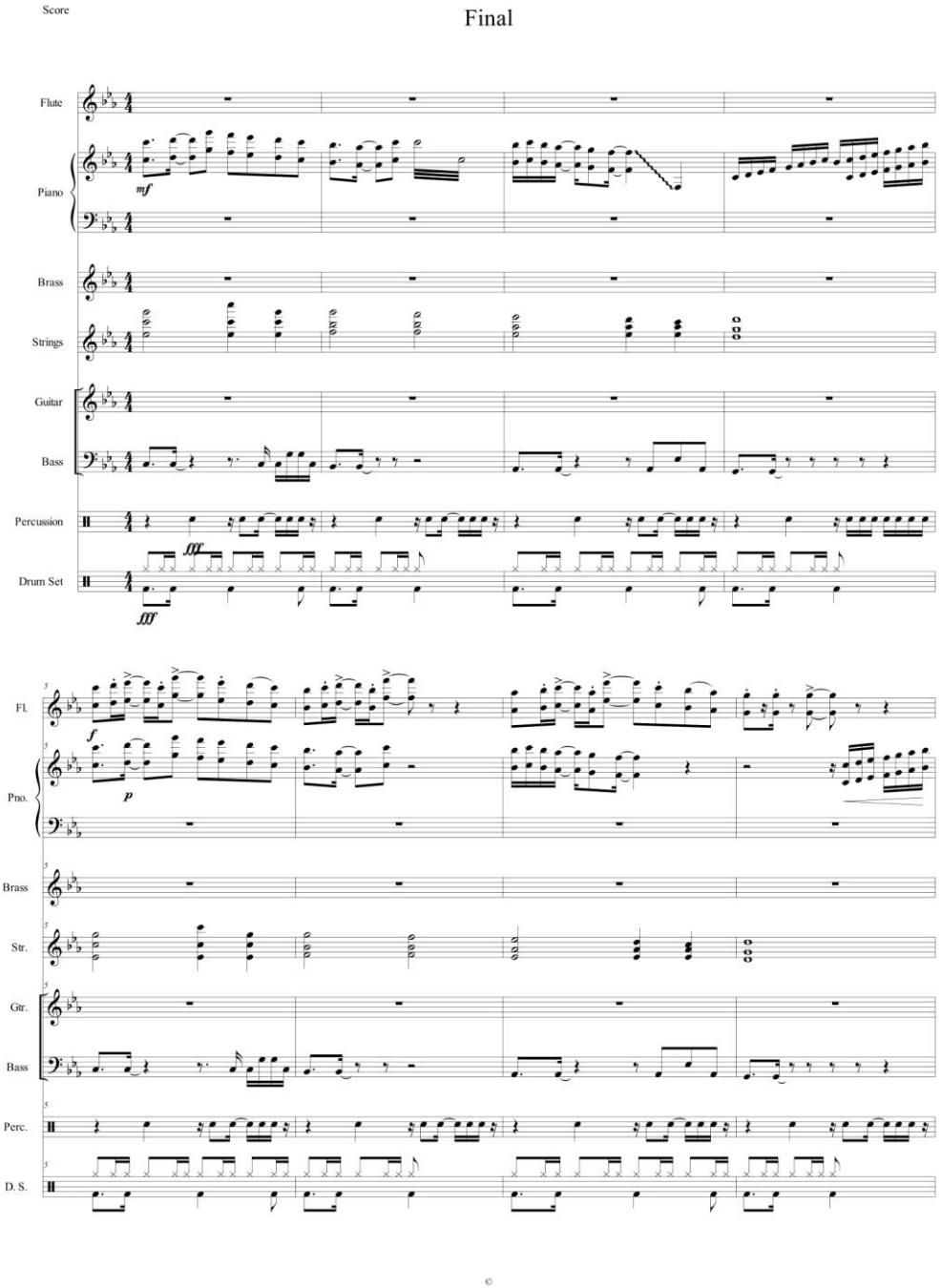
Gel

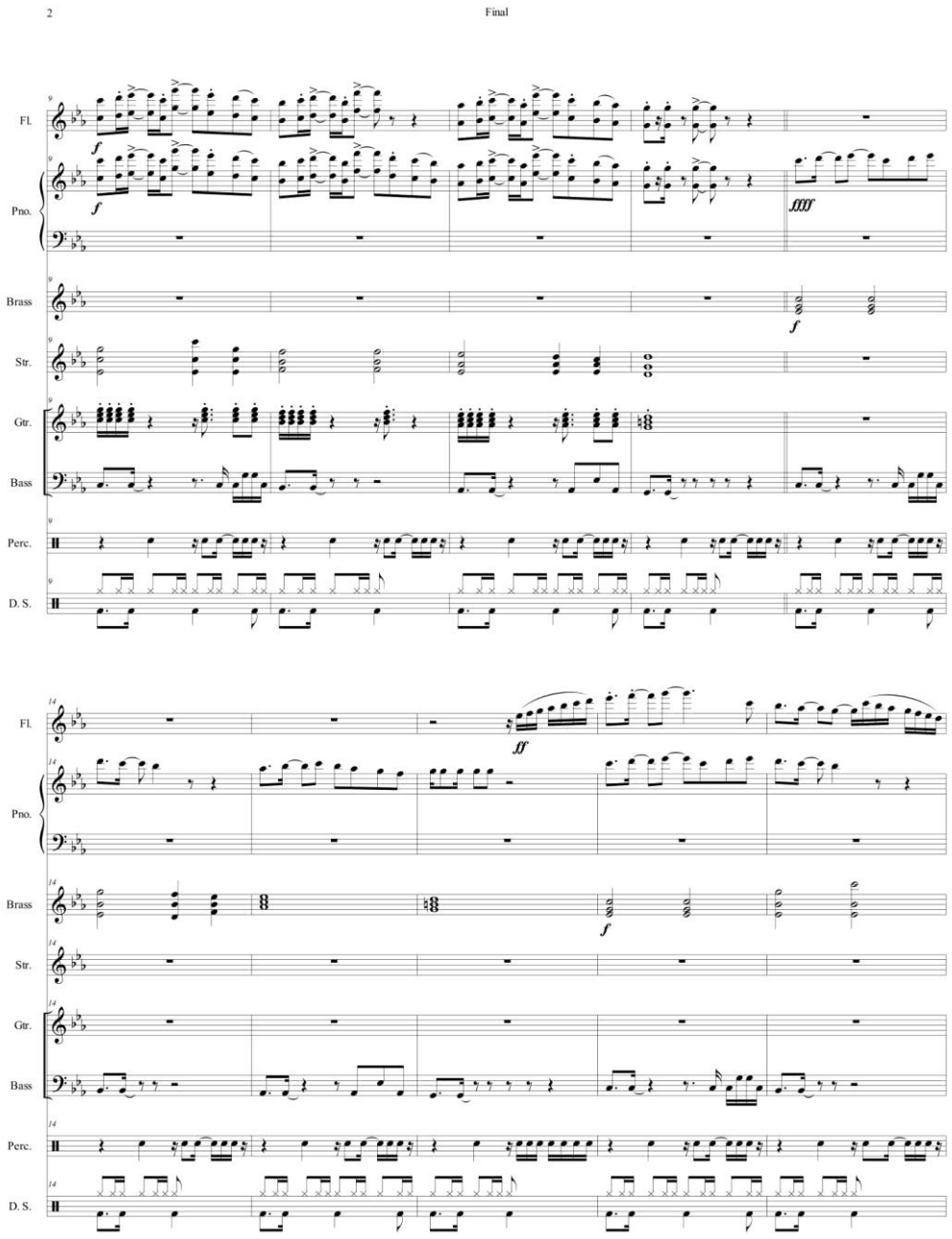

Ice
Gerard Queralt
♩ = 92
Horn in F
Choir
Choir
Strings I
Strings II
Strings III
Contrabass
Timpani
Hn.
C
C
Str. I
Str. II
Str. III
Cb.
Timp.

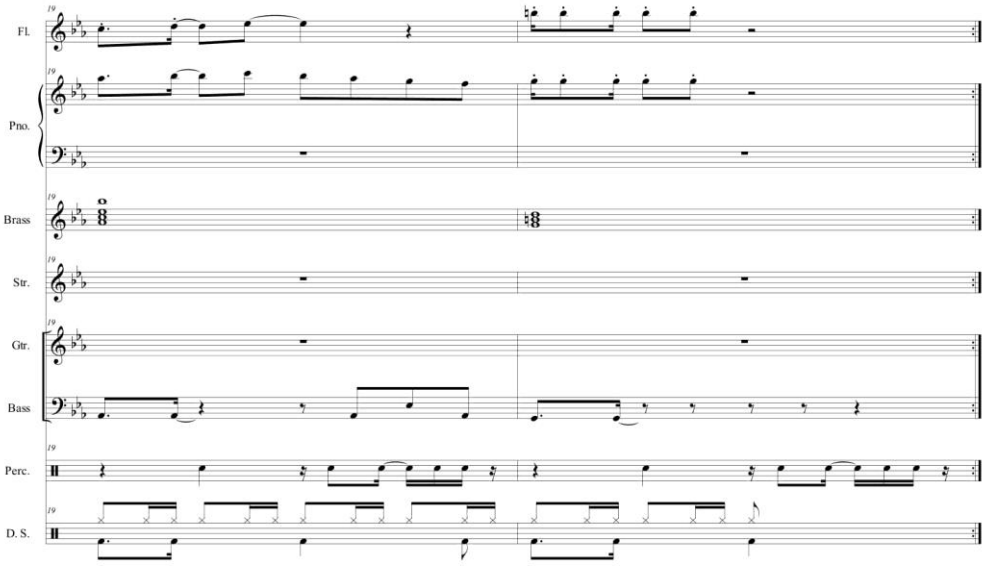
Castle


Castle
Gerard Queralt
Maestoso
Horn in E♭
Piano
Trumpet in C
Strings
Bass
Percussion
Drum Set
Hn.
Pno.
C Tpt.
Str.
Bass
Perc.
D. S.

“Boss” Final


Score
Final
Flute
Piano
Brass
Strings
Guitar
Bass
Percussion
Drum Set
Fl.
Pno.
Brass
Str.
Gtr.
Bass
Perc.
D. S.


2
Final
Fl.
Pno.
Brass
Str.
Gtr.
Bass
Perc.
D. S.
f
ff


19
Fl.
Pno.
Brass
Str.
Gtr.
Bass
Perc.
D. S.

Webgrafia

Unity

Unity (Consultada 12-6-2017) https://unity3d.com/

Assets

Custom Character 2D Vol. 1 (Consultada 20-7-2017)
https://www.assetstore.unity3d.com/en/#!/content/65144

Custom Character 2D Vol. 3 (Consultada 20-7-2017)
https://www.assetstore.unity3d.com/en/#!/content/65471

2D Jungle Side-Scrolling Plataformer Pack (Consultada 30-8-2017)
https://www.assetstore.unity3d.com/en/#!/content/78506

2D Volcano Side-Scrolling Platformer Pack (Consultada 30-8-2017)
https://www.assetstore.unity3d.com/en/#!/content/79184

2D Northland Side-Scrolling Platformer Pack (Consultada 30-8-2017)
https://www.assetstore.unity3d.com/en/#!/content/78545

904 HQ Hand Painted Bundle Fantasy Medieval (Consultada 30-8-2017)
https://www.assetstore.unity3d.com/en/#!/content/86574

Sunset Landscape (Consultada 22-12-2017) https://www.pexels.com/photo/sunset-landscape-111755/

Meath Font (Consultada 22-12-2017) https://www.1001freefonts.com/meath.font

Hearts.png (Consultada 17-12-2017)
https://drive.google.com/file/d/0B0osKu5IdBZgMS1pNEkyNE1fQVU/view

Tutorials

Roll-a-ball Tutorial (Consultada 25-6-2017) https://unity3d.com/learn/tutorials/s/roll-ball-tutorial

2D Game Creation (Consultada 16-8-2017) https://unity3d.com/learn/tutorials/s/2d-game-creation

Unity 5 2D Platformer Tutorial - Part 7 - Smooth Camera Movement (Consultada 14-10-2017) https://www.youtube.com/watch?v=BQEsbOALKhc

Unity 5 2D Platformer Tutorial - Part 8 - Camera Bounds (Consultada 14-10-2017) https://www.youtube.com/watch?v=I6xmOMsRWeo

Creating a Start Menu / Main Menu in Unity 5 (Consultada 20-10-2017) https://www.youtube.com/watch?v=FrJogRBSzFo

Unity 5 2D Platformer Tutorial - Part 10 - Creating a Pause Menu (part 1) (Consultada 21-10-2017) https://www.youtube.com/watch?v=xIevsYimJYc

Unity 5 2D Platformer Tutorial - Part 11 - Creating a Pause Menu (part 2) (Consultada 21-10-2017) https://www.youtube.com/watch?v=TatAnGj1RMg

Checkpoints & Respawning - Unity 2D Platformer Tutorial - Part 3 (Consultada 22-11-2017) https://www.youtube.com/watch?v=ndYd4S7UkAU

Unity 5 2D Platformer Tutorial - Part 12 - Player Health (Consultada 16-12-2017) https://www.youtube.com/watch?v=l6rOHJitheY

Unity 5 2D Platformer Tutorial - Part 13 - Player Health UI (Consultada 17-12-2017) https://www.youtube.com/watch?v=5KwkfGfaRNU

Knockback & Stomping Enemies - Unity 2D Platformer Tutorial - Part 11 (Consultada 3-1-2018) https://www.youtube.com/watch?v=sdGeGQPPW7E

Adding A Health System - Unity 2D Platformer Tutorial - Part 9 (Consultada 3-1-2018) https://www.youtube.com/watch?v=F6hUIU72JwE

Unity 5 2D Platformer Tutorial - Part 14 - Spikes (Consultada 3-1-2018) https://www.youtube.com/watch?v=YLfII90PA-c

Conceptes

Unity (Game engine) (Consultada 5-12-2017) https://en.wikipedia.org/wiki/Unity_(game_engine)

C# (Programming language) (Consultada 5-12-2017) https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Finale (Software) (Consultada 5-12-2017) https://en.wikipedia.org/wiki/Finale_(software)