Mancarella Chiara

Relazione di laboratorio: realizzazione di un sito web

Space Junk

Introduzione

Gli argomenti su cui si basa questo sito sono lo spazio, l'intelligenza artificiale e i detriti spaziali (che a differenza di quanto si crede sono numerosi).
Perchè ho scelto questo tema? Perché è l'unione di due miei grandi interessi: l'intelligenza artificiale e lo spazio.

Oggetto e note

Per visualizzare il sito cliccare su questo link (https://chiara2804.github.io/Mancarella_Chiara-Space_Junk/).

Inoltre è consigliato (non necessario) utilizzare il browser di Chrome, uno schermo con risoluzione 1920 x 1080 e con ratio 16:9.

Linguaggi:

- HTML 5;
- CSS3;
- JavaScript.

Framework e librerie:

- A-FRAME, framework basato sulla libreria Three.js;
- jQuery (versione `3.3.1`);
- Cesium, una libreria open source utilizzata per creare globi 3D e mappe;
- Satellite, una libreria per rendere possibile la diffusione satellitare tramite TLE nel web. Fornisce le funzioni necessarie per i calcoli SGP4/SDP4. Fornisce anche funzioni per le trasformazioni di coordinate;
- Leaflet, una libreria per sviluppare mappe geografiche interattive;
- Form Spree, è un servizio di backend di moduli, API e e-mail per moduli HTML e JavaScript.

Organizzazione file e cartelle

La struttura delle directory è rappresentata nel file "Organizzazione file e cartelle.pdf".
In breve, si è deciso di dividere le immagini (imgs) dalle pagine (pgs) e dai file sonori (sound). A loro volta esse sono divise in altre sottocartelle corrispondenti al nome delle pagine HTML, al fine di avere una divisione chiara e ordinata.

Caratteristiche generali

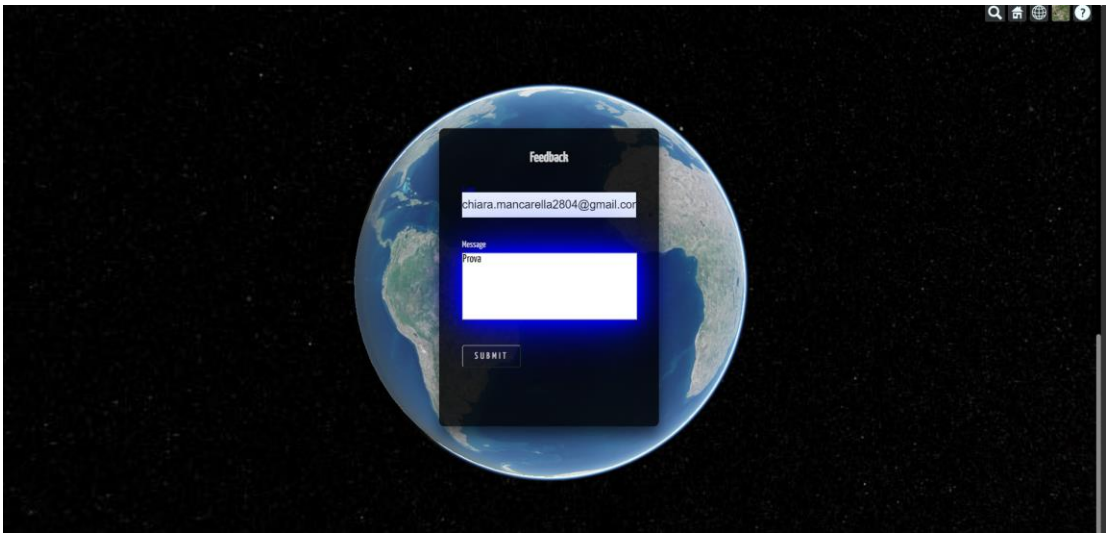
Il sito è stato realizzato completamente in lingua inglese.
I brani musicali utilizzati sono senza Copyright.

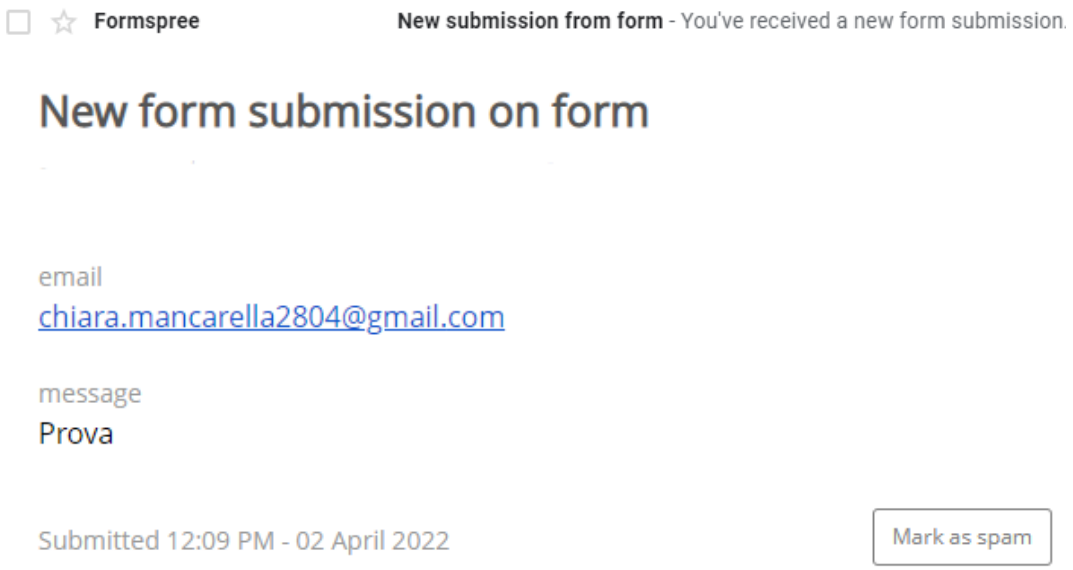
In ogni tag <head> di pagina è stato incluso il link dell'API di Google Font e l'icona della pagina visibile nella tab, utilizzando:

```
<link rel="icon" href="../../imgs/General/icons8-earth-64.png">
```

Nel tag <body> sono presenti un context menu e scrollbars personalizzati, dei bottoni per tornare alla pagina precedente e il tasto per inviare un feedback (icona dell'astronauta in alto a destra).
In particolare:

- Per i feedback è stato usato Form Spree, il quale permette di inviare e ricevere e-mail senza dover creare un server locale.





- Per le barre di scorrimento è stato usato il motore di rendering per browser web `webkit`.

Prima della chiusura di ogni tag <body> sono inclusi i file JavaScript utilizzati, essi non sono stati posti nel tag <head> al fine di avere un caricamento più veloce.

Sono state utilizzate unità di misura relative come vw e vh, il modello di layout web, comunemente noto come Flexbox, e i CSS Media Queries (@media) con lo scopo di rendere le pagine responsive.

- ***index.html***

Oggetto: mostrare le istruzioni per muoversi all'interno della pagina principale. Collegamento a pgs/pgs/index1.html.

La pagina è caratterizzata dall'effetto estetico di typing del testo.:

Spiegazione (pgs/script/script-introAnimation/script-timedRedirection.js):
Viene inizializzato un array di stringhe contenenti il testo da mostrare:

```
var aText = new Array('[text]', '[text]', ...);
```

Definizione di alcune costanti:

```
var iSpeed = 50; // tempo di delay tra una lettera e l'altra
var iIndex = 0; // parto da questa posizione
var iArrLength = aText[0].length; // lunghezza dell'array
var iScrollAt = 20; // si abilita lo scroll dopo 20 righe

var iTextPos = 0; // posizione del testo
var sContents = '';
var iRow; // riga corrente
```

La funzione typewriter() stampa ogni lettera con un delay definito dalla variabile iSpeed, quando finisce la stringa avrà un delay maggiore definito dal metodo setTimeout().
Alla fine dell'ultima stringa indirizzerà la pagina a pgs/pgs/index1.html dopo 6 secondi grazie al metodo setTimeout().

```
function typewriter() {
    sContents = ' ';
    iRow = Math.max(0, iIndex - iScrollAt);
    var destination = document.getElementById("text");

    while (iRow < iIndex)
        sContents += aText[iRow++] + '<br />';
  destination.innerHTML   =   sContents   +   aText[iIndex].substring(0,
iTextPos);
    if (iTextPos++ == iArrLength) {
        iTextPos = 0;
        iIndex++;
        if (iIndex != aText.length) {
            iArrLength = aText[iIndex].length;
            setTimeout("typewriter()", 1500);
        } else if (iIndex == aText.length) {
            setTimeout(function() {
                window.location.href = 'pgs/pgs/index1.html';
            }, 6000);
        }
    } else
```

```
        setTimeout("typewriter()", iSpeed);
}
```

- ***pgs/pgs/index1.html***

Oggetto: pagina principale da cui collegarsi alle altre pagine.
Viene utilizzato il framework A-FRAME basato sulla libreria Three.js nata per creare e visualizzare grafica computerizzata 3D animata in un browser Web utilizzando WebGL.

Inizialmente appare un'animazione di caricamento, grazie alla quale la parte che utilizza il framework ha tempo di caricarsi e di visualizzarsi correttamente.

L'animazione è completamente realizzata in CSS, mediante l'uso di @keyframes che permette di passare gradualmente da un insieme di stili CSS all'altro.
Durante l'animazione, si può modificare più volte l'insieme degli stili CSS, infatti si deve specificare quando avverrà il cambio di stile in percentuale, o con le parole chiave "from" e "to", che sono uguali a 0% e 100%. 0% è l'inizio dell'animazione, 100% è quando l'animazione è completa.

Esempio di una parte dell'animazione (pgs/style/introAnimation/style-introAnimation.html):

```
.orbit1 {
    animation: orbit1 infinite 3s linear;
}
```

Alla classe .orbit1 viene assegnata l'animazione "orbit1" con ripetizione infinita, tempo di completamento di 3 secondi e velocità costante.

```
@keyframes orbit1 {
    from {
        transform: translateZ(1px) rotate3d(2, 1, 0, 0deg) rotate(0deg);
    } to {
        transform:   translateZ(1px)   rotate3d(2,   1,   0,   -360deg)
rotate(720deg);
    }
}
```

L'animazione "orbit1", utilizzando la funzione CSS translateZ(), sposta di 1px il cerchio nell'asse z.
La funzione CSS rotate3d(), definisce una trasformazione che ruota un elemento attorno ad un asse fisso nello spazio 3D, senza deformarlo.
La funzione CSS rotate(), definisce una trasformazione che ruota un elemento attorno ad un punto fisso sul piano 2D, senza deformarlo.

Finita l'animazione, viene visualizzata la propriamente detta pagina principale, realizzata mediante il framework sopra citato..
Presenta uno sfondo a 360° in cui è possibile muoversi come nella realtà aumentata.

La pagina è composta da quattro cerchi, grazie ai quali, cliccando al loro interno, è possibile collegarsi alle altre pagine: pgs/pgs/ISSConnection.html, pgs/pgs/mappingSpaceDebris.html, pgs/pgs/AIInSpace.html, pgs/pgs/launches.html. Inoltre, è possibile rendere l'esperienza più

coinvolgente cliccando sul tasto centrale per avviare la musica e sul tasto in basso a destra per attivare la funzione schermo intero.

La scena è creata da un tag <a-scene> al cui interno:

```
<a-scene id="vr">
     <a-sky src="../../imgs/index/background.jpg">
         <a-animation attribute="..." fill="..." easing="..." dur="..."
from="..." to="..." repeat="..."></a-animation>
     </a-sky>

     <a-camera position="..." id="camera">
  <a-entity       cursor="..."       position="..."       geometry="..."
material="..."></a-entity>
     </a-camera>

     <a-text font="..." position="..." rotation="...">
           <a-link  href="AIInSpace.html"  title="..."  image="#bg"></a-
link>
     </a-text>

     <a-assets>
 <audio id="sound_f" src="../../sound/Sound.mp3" preload="auto"></audio>
     </a-assets>
           <a-entity sound_vr text="..." geometry="..." position="..."
material="..."></a-entity>
</a-scene>
```

- <a-scene> costituisce l'immagine di sfondo, a cui, in questo caso, è stata attribuita un'animazione tramite il tag <a-animation>.
- <a-camera> definisce la posizione di vista iniziale.
- <a-entity>, in questo caso è utilizzato per creare il puntatore (un cerchio semi-trasparente) e il box per avviare e fermare la musica.

  Per rendere possibile quest'ultima è stato necessario usare uno script (pgs/script/script-index/script-playSound.js) che, selezionando l'id del file audio, riproduce il suono in base al click del mouse e al valore della variabile playing:
  - se playing == false l'audio non verrà riprodotto;
  - se playing != false l'audio verrà riprodotto.

```
AFRAME.registerComponent('sound_vr', {
    init: function() {
        let playing = false;
        let audio = document.querySelector("#sound_f");
        this.el.addEventListener('click', () => {
            if (!playing)
                audio.play();
            else
```

```
                audio.pause();
            playing = !playing;
        }); } })
```

- <a-text> è combinato al tag <a-link> che consente di collegarsi alle altre pagine.

- pgs/pgs/ISSConnection, pgs/pgs/ISSMap

Oggetto: visualizzare la posizione e la vista dalla Stazione Spaziale Internazionale in tempo reale.

La pagina pgs/pgs/ISSConnection.html è composta principalmente da due tag <iframe>.

- Nel primo si visualizza un video live da YouTube.
- Nel secondo si visualizza la pagina pgs/pgs/ISSMap.html.
  Essa mostra la posizione attuale della ISS sulla cartina e i relativi valori di velocità, posizione e altitudine.
  La cartina è stata creata grazie all'utilizzo di Leaflet.

  Per visualizzare l'icona della ISS nella posizione corretta sono state usate due funzioni contenute all'interno del file pgs/script/script-ISSMap/script-ISSCurrentPosition.js.

  La prima, findISS(), tramite un'API ottiene i dati relativi a posizione, latitudine, altitudine, velocità e visibilità (notte/giorno) ogni due secondi. A sua volta chiama la funzione updateISS(), a cui passa i valori ricevuti per aggiornarli all'interno del box delle informazioni e sulla cartina.

```
function findISS() {
    fetch("https://api.wheretheiss.at/v1/satellites/25544")
        .then(response => response.json())
        .then(data => {
            lat = data.latitude.toFixed(2);
            long = data.longitude.toFixed(2);
       const timestamp = new Date(data.timestamp * 1000).toUTCString();
            const speed = data.velocity.toFixed(2);
            const altitude = data.altitude.toFixed(2);
            const visibility = data.visibility;
           updateISS(lat, long, timestamp, speed, altitude, visibility);
        })
        .catch(e => console.log(e));
}
```

- pgs/pgs/AIInSpace.html

Oggetto: descrivere le funzioni che l'intelligenza artificiale svolge in ambito spaziale. Collegamento alle pagine pgs/pgs/CIMON.html, pgs/pgs/ProcessingSat.html, pgs/pgs/roleAI.html.

Effetto di parallasse (pgs/style/AIInSpace/parallax.css):

```
background-attachment: fixed;
```

```
background-position: center;
background-repeat: no-repeat;
background-size: cover;
```

Per trattare i diversi argomenti è stato creato un riquadro che contiene a sua volta tre riquadri più piccoli. Esso è caratterizzato da un effetto estetico particolare: è sensibile alla posizione del mouse e si sposta in base ad essa come se venisse spinto.

Spiegazione (pgs/script/script-AIInSpace/script-cards.js):
Il metodo document.querySelector() accede, in lettura e scrittura, all'elemento HTML della pagina tramite i selettori CSS.
calcValue è la costante che definisce di quanto il riquadro si deve spostare al passaggio del mouse.

```
const cards = document.querySelector(".cards");
const images = document.querySelectorAll(".card__img");
const backgrounds = document.querySelectorAll(".card__bg");
const range = 40;

const calcValue = (a, b) => (a / b * range - range / 2).toFixed(1)
```

Il metodo document.addEventListener() rileva il movimento del mouse negli assi x e y del documento:

```
document.addEventListener('mousemove', ({ x, y })
```

Calcolo di quanto si sposta il mouse:

```
timeout = window.requestAnimationFrame(() => {
        const yValue = calcValue(y, window.innerHeight);
        const xValue = calcValue(x, window.innerWidth);

        cards.style.transform          =          `rotateX(${yValue}deg)
rotateY(${xValue}deg)`;
```

Per ogni chiamata sposto i riquadri dei valori calcolati precedentemente modificando le proprietà di stile degli elementi:

```
[].forEach.call(images, (image) => {
      image.style.transform          =          `translateX(${-xValue}px)
translateY(${yValue}px)`; });

[].forEach.call(backgrounds, (background) => {
 background.style.backgroundPosition      =      `${xValue*.45}px      ${-
yValue*.45}px`; })
```

- ***pgs/pgs/CIMON.html, pgs/pgs/ProcessingSat.html, pgs/pgs/roleAI.html***

- Oggetto pgs/pgs/CIMON.html: mostrare in particolare il ruolo di CIMON (un assistente artificiale). È inoltre disponibile la mappa sensibile con approfondimento sui suoi componenti.
- Oggetto pgs/pgs/ProcessingSat.html: descrivere l'importanza e l'utilizzo dei dati processati da intelligenze artificiali nello spazio e sulla Terra.
- Oggetto pgs/pgs/roleAI.html: spiegare che l'intelligenza artificiale è usata anche per l'assemblaggio dei velivoli spaziali allo scopo di eliminare la contaminazione biologica.

In ognuna di queste pagine il titolo è reso con un effetto glitch creato utilizzando @keyframe.

- ***pgs/pgs/CIMON.html***

La mappa sensibile è sopra l'immagine di CIMON. Affinché anch'essa fosse responsive si sono utilizzati i tag <svg> (Scalable Vector Graphics).

Esempio di una parte di mappa sensibile:

```
<svg xmlns="http://www.w3.org/2000/svg" fill-opacity="1">
  <a href="https://en.wikipedia.org/wiki/Digital_camera" target="_new">
      <circle cx="23vw" cy="5.8vw" r="1vw" fill-opacity="0"/>
  </a>
</svg>
```

(pgs/style/AIInSpace/style-map.css)

```
.img1 {
    width: 50vw;
    height: 30vw;
    background-image: url('../../../imgs/AIInSpace/CIMON_x2.jpg');
    margin: 2.5vw 0 1vw 0;
    background-size: cover;
}
```

```
svg {
    width: 50vw;
    height: 30vw;
}
```

- ***pgs/pgs/launches.html***

Oggetto: mostrare un globo che indica alcuni dei principali punti di lancio dei razzi o satelliti artificiali.

Creazione di un globo (pgs/script/mappingSpaceDebris/script-globe.js):

```
var viewer = new Cesium.Viewer("cesiumContainer", {
    timeline: false,
```

```
    animation: false,
});
var scene = viewer.scene;
var primitives = scene.primitives;
```

In questo caso si è voluta disattivare anche la barra del tempo che solitamente compare nella parte inferiore del contenitore Cesium.

Aggiunta di un simbolo che indichi un punto di lancio:

```
var instance0 = new Cesium.GeometryInstance({
    geometry: new Cesium.CircleGeometry({
        center: Cesium.Cartesian3.fromDegrees([long], [lat]),
        radius: [value]
    }),
    id: 'KSC',
    attributes: {
        color:
Cesium.ColorGeometryInstanceAttribute.fromColor(Cesium.Color.RED)
    }
});

scene.primitives.add(new Cesium.Primitive({
    geometryInstances: instance0,
    appearance: new Cesium.PerInstanceColorAppearance({
        flat: true,
        translucent: false
    })
}));
```

Ogni variabile instance rappresenta un cerchio di colore rosso posto nel posto in cui c'è un punto di lancio (le coordinate sono fornite dai valori [lat] e [long]).

Al click dell'utente sopra un qualsiasi cerchio rosso, compare un riquadro con delle informazioni riguardo al punto selezionato. Per fare ciò si utilizza un metodo che tramite l'id setta l'attributo opacity a 0 o a 1.

```
var handler = new Cesium.ScreenSpaceEventHandler(scene.canvas);

handler.setInputAction(function(movement) {
    var pick = scene.pick(movement.position);
    if (Cesium.defined(pick) && (pick.id === 'KSC')) {
        document.getElementById('KSC').style.opacity = 1;
        document.getElementById('[other]').style.opacity = 0;
}, Cesium.ScreenSpaceEventType.LEFT_CLICK);
```

- ***pgs/pgs/mappingSpaceDebris.html***

Oggetto: mostrare un globo con i detriti in orbita. Collegamento a pgs/pgs/game.html.

La rappresentazione dei detriti è stata resa grazie alla libreria Satellite.
I dati TLE utilizzati sono stati presi dal sito di Celestrak.

Spiegazione (pgs/script/script-mappingSpaceDebris/script-globe.js):
Dichiarazione un vettore in cui ogni elemento è una stringa che rappresenta una riga del dato TLE:

```
var debr = [
    "1 24946U 97051C    18072.81969124   .00000077   00000-0   20774-4 0
9993",
    "2 24946   86.3936 173.9133 0010038   87.3628 272.8716 14.33594224
72655", . . . ];
```

Il frammento di codice seguente scorre ogni TLE e aggiunge un'entità punto a una matrice. Il contatore del loop diventa l'id per il punto e un indice nell'array.

```
var thing = [];
for (debrisID = 0; debrisID < debr.length; debrisID++) {
    thing[debrisID] = viewer.entities.add({
        position: {
            value: Cesium.Cartesian3.fromDegrees(-75.59777, 40.03883),
            referenceFrame: Cesium.ReferenceFrame.FIXED
        },
        point: {
            color: Cesium.Color.ORANGE,
            pixelSize: 4
        }
    });
}
```

Le funzioni all'interno della libreria Satellite leggono il TLE e creano un record, propagano la posizione in base all'avanzamento del tempo e generano coordinate cartesiane in un sistema di riferimento Earth Centered Fixed o Earth Centered Inertial. Il frammento di codice seguente genera una matrice di record dai dati TLE, genera una matrice di posizioni e velocità.

```
var debrisRecords = [];
var datasetSize = debr.length;
console.log(datasetSize);
var posVel = [];

function propagateOrbitalDebris() {
    var j = 0;
    for (i = 0; i < datasetSize; i++) {
        var tle1 = debr[j];
        var tle2 = debr[j + 1];
        if (typeof tle1 == 'string' || tle1 instanceof String || typeof
tle2 == 'string' || tle2 instanceof String) {
```

```
            debrisRecords[i] = satellite.twoline2satrec(tle1, tle2);
        }
        j = j + 2;
    }
    // Propagate debris using time since epoch
    for (i = 0; i < datasetSize; i++) {
        if (debrisRecords[i] != undefined) {
            posVel[i]         =         satellite.sgp4(debrisRecords[i],
timeSinceTleEpochMinutes);
        }
    }
    // Propagate debris using time since epoch
    for (i = 0; i < datasetSize; i++) {
        if (debrisRecords[i] != undefined) {
            posVel[i] = satellite.propagate(
                debrisRecords[i],
                now.getUTCFullYear(),
                now.getUTCMonth() + 1,
                now.getUTCDate(),
                now.getUTCHours(),
                now.getUTCMinutes(),
                now.getUTCSeconds()
            );
        } } }
```

- ***pgs/pgs/game.html***

Oggetto: gioco in cui bisogna distruggere i detriti.

Definisco i tasti freccia verso sinistra, freccia verso destra e il tasto Q e il rispettivo comportamento se premuti (pgs/script/game/app.js).
Il tasto Q, se premuto, esegue la funzione di sparare un proiettile e incrementa i punti ogni volta che il proiettile si trova nella stessa posizione della roccia.

Genero dei punti con un valore tra 0 e 470 in cui far comparire la roccia ogni 1700 millisecondi:

```
var generaterocks = setInterval(() => {
    var rock = document.createElement("div");
    rock.classList.add("rocks");
    var rockleft = parseInt(
        window.getComputedStyle(rock).getPropertyValue("left")
    );
    rock.style.left = Math.floor(Math.random() * 470) + "px";
```

```
    board.appendChild(rock);
}, 1700);
```

Faccio muovere la roccia generata fino a un'altezza di 500px verso il basso:

```
var moverocks = setInterval(() => {
    var rocks = document.getElementsByClassName("rocks");

    if (rocks != undefined) {
        for (var i = 0; i < rocks.length; i++) {
            var rock = rocks[i];
            var rocktop = parseInt(
                window.getComputedStyle(rock).getPropertyValue("top")
            );
```

Se la roccia raggiunge quell'altezza prima di essere distrutta, il gioco viene fermato:

```
            if (rocktop >= 475) {
                m.pause();
                alert('Game Over');
                clearInterval(moverocks);
                window.location.reload();
            }
            rock.style.top = rocktop + 25 + "px";
        }
    }
}, 500);
```