--- language.js ---

//languageData.js
export const languageData = {
vi: {
title: "Game Hứng Táo",
startGame: "Chơi ngay",
settings: "Cài đặt",
fallSpeed: "Tốc độ rơi:",
basketSpeed: "Tốc độ di chuyển giỏ:",
controlType: "Chọn kiểu điều khiển:",
keyboard: "Bàn phím",
handtracking: "Hand Tracking",
languageSelect: "Chọn ngôn ngữ:",
save: "Lưu",
back: "Quay lại",
returnMenu: "Về Menu",
score: "Điểm",
time: "Thời gian",
control: "Điều khiển"
},
en: {
title: "Apple Catch Game",
startGame: "Play Now",
settings: "Settings",
fallSpeed: "Fall Speed:",
basketSpeed: "Basket Move Speed:",
controlType: "Select Control Type:",
keyboard: "Keyboard",
handtracking: "Hand Tracking",
languageSelect: "Select Language:",
save: "Save",
back: "Back",
returnMenu: "Return to Menu",
score: "Score",
time: "Time",
control: "Control"
},
zh: {
title: "接苹果游戏",
startGame: "开始游戏",
settings: "设置",
fallSpeed: "掉落速度:",
basketSpeed: "篮子移动速度:",
controlType: "选择控制方式:",
keyboard: "键盘",
handtracking: "手部追踪",
languageSelect: "选择语言:",
save: "保存",
back: "返回",
returnMenu: "返回菜单",
score: "得分",
time: "时间",
control: "控制方式"
}
};

--- handsModule.js ---

// handsModule.js
export const setupHandsTracking = (canvasElement, toggleBtn) => {
const canvasCtx = canvasElement.getContext("2d");
const hands = new Hands({
locateFile: (file) => `https://cdn.jsdelivr.net/npm/@mediapipe/hands/${file}`
});
hands.setOptions({
maxNumHands: 2,
modelComplexity: 1,
minDetectionConfidence: 0.5,
minTrackingConfidence: 0.5
});
hands.onResults((results) => {
canvasCtx.clearRect(0, 0, canvasElement.width, canvasElement.height);
if (results.multiHandLandmarks) {
results.multiHandLandmarks.forEach((landmarks) => {
currentHandX = landmarks[9].x; // lấy tọa độ X của điểm lòng bàn tay
canvasCtx.strokeStyle = "yellow";
canvasCtx.lineWidth = 2;
for (let i = 0; i < landmarks.length - 1; i++) {
canvasCtx.beginPath();
canvasCtx.moveTo(landmarks[i].x \* canvasElement.width, landmarks[i].y \* canvasElement.height);
canvasCtx.lineTo(landmarks[i + 1].x \* canvasElement.width, landmarks[i + 1].y \* canvasElement.height);
canvasCtx.stroke();
}
// Hiển thị từng điểm landmark nhỏ màu đỏ
canvasCtx.fillStyle = "red";
landmarks.forEach((landmark) => {
canvasCtx.beginPath();
canvasCtx.arc(landmark.x \* canvasElement.width, landmark.y \* canvasElement.height, 2, 0, 2 \* Math.PI);
canvasCtx.fill();
});
});
}
});
return hands;
};
export const setupWebcamToggle = (toggleBtn, hands) => {
let streamActive = false;
let camera;
toggleBtn.addEventListener("click", () => {
if (!streamActive) {
const videoElement = document.createElement("video");
videoElement.width = 640;
videoElement.height = 480;
camera = new Camera(videoElement, {
onFrame: async () => await hands.send({ image: videoElement }),
width: 640,
height: 480
});
camera.start();
toggleBtn.textContent = "Tắt Webcam";
toggleBtn.classList.add("active"); // Đổi màu sang đỏ khi bật
console.log("Webcam BẬT - Đã thêm class 'active'");
streamActive = true;
} else {
camera.stop();
toggleBtn.textContent = "Bật Webcam";
toggleBtn.classList.remove("active"); // Đổi màu sang trắng khi tắt
console.log("Webcam TẮT - Đã xóa class 'active'");
streamActive = false;
}
});
};
let currentHandX = 0;
export const getCurrentHandX = () => currentHandX;

--- gamescene.js ---

//gamescene.js
import { getCurrentHandX } from './handsModule.js';
import { languageData } from './language.js';
export class GameScene extends Phaser.Scene {
constructor() {
super("scene-game");
this.player;
this.target;
this.cursor;
this.score = 0;
this.timeElapsed = 0;
this.scoreText;
this.timeText;
this.controlText;
this.mode = localStorage.getItem("gameMode") || "free";
this.remainingTime = 60; // 60s nếu cần
this.timerText; // sẽ tạo 1 text hiện thời gian đếm ngược
this.gameEnded = false; // kiểm soát trạng thái kết thúc
}
preload() {
this.load.image("bg", "public/assets/caytao.png");
this.load.image("basket", "public/assets/basket.png");
this.load.image("apple", "public/assets/apple.png");
// Âm thanh game
this.load.audio("bgMusic", "public/assets/bgMusic.mp3");
this.load.audio("coinSound", "public/assets/coin.mp3");
}
create() {
//Âm thanh background
this.bgMusic = this.sound.add("bgMusic", { loop: true, volume: 0.5 });
this.bgMusic.play();
if (this.mode === "60s") {
this.timerText = this.add.text(300, 10, "60", { fontSize: "22px", fill: "#ff3333", stroke: "#000", strokeThickness: 2 });
}
const lang = localStorage.getItem("language") || "vi";
const texts = languageData[lang];
this.add.image(0, 0, "bg").setOrigin(0, 0);
this.player = this.physics.add.image(0, 500, "basket").setCollideWorldBounds(true);
this.target = this.physics.add.image(Math.random() \* 500, 0, "apple").setVelocityY(50);
this.cursor = this.input.keyboard.createCursorKeys();
this.scoreText = this.add.text(10, 10, `${texts.score}: 0`, { fontSize: "22px", fill: "#ffcc00", stroke: "#000", strokeThickness: 2 });
this.timeText = this.add.text(10, 40, `${texts.time}: 0s`, { fontSize: "20px", fill: "#66ff99", stroke: "#000", strokeThickness: 2 });
this.controlText = this.add.text(10, 70, "", { fontSize: "18px", fill: "#ffffff", strokeThickness: 1 });
this.modeText = this.add.text(10, 100, "", { fontSize: "18px", fill: "#ffffff", strokeThickness: 1 });
this.physics.add.overlap(this.player, this.target, this.catchApple, null, this);
}
update() {
const lang = localStorage.getItem("language") || "vi";
const texts = languageData[lang];
const savedControlType = localStorage.getItem("controlType") || "keyboard";
this.controlText.setText(`${texts.control}: ${savedControlType === "keyboard" ? texts.keyboard : texts.handtracking}`);
this.modeText.setText(`Chế độ: ${this.mode === "60s" ? "60 Giây" : "Tự Do"}`);
if (this.mode === "60s" && !this.gameEnded) {
this.remainingTime -= this.game.loop.delta / 1000;
if (this.remainingTime <= 0) {
this.remainingTime = 0;
this.endGame();
}
this.timeText.setText(`${texts.time}: ${Math.floor(this.remainingTime)}s`);
} else {
this.timeElapsed += this.game.loop.delta / 1000;
this.timeText.setText(`${texts.time}: ${Math.floor(this.timeElapsed)}s`);
}
const controlType = localStorage.getItem("controlType") || "keyboard";
if (controlType === "handtracking") {
const handX = getCurrentHandX(); // [0, 1] là tỉ lệ so với khung hình webcam
const canvasWidth = this.sys.canvas.width;
const targetX = handX \* canvasWidth;
this.player.x = Phaser.Math.Clamp(targetX, 0 + this.player.width / 2, canvasWidth - this.player.width / 2);
this.player.setVelocityX(0); // giữ đứng yên theo trục X vì ta cập nhật trực tiếp
} else {
if (this.cursor.left.isDown) {
this.player.setVelocityX(-200);
} else if (this.cursor.right.isDown) {
this.player.setVelocityX(200);
} else {
this.player.setVelocityX(0);
}
}
if (this.target.y >= 500) {
this.target.setY(0);
this.target.setX(Math.random() \* 500);
}
}
catchApple() {
//Âm thanh khi bắt táo
this.sound.play("coinSound", { volume: 1 });
this.score += 10;
this.scoreText.setText(`${languageData.vi.score}: ${this.score}`);
this.target.setY(0);
this.target.setX(Math.random() \* 500);
const lang = localStorage.getItem("language") || "vi";
const texts = languageData[lang];
this.score += 10;
this.scoreText.setText(`${texts.score}: ${this.score}`);
this.target.setY(0);
this.target.setX(Math.random() \* 500);
}
endGame() {
this.gameEnded = true;
this.physics.pause();
const lang = localStorage.getItem("language") || "vi";
const texts = languageData[lang];
this.add.text(150, 220, `Kết thúc!\n${texts.score}: ${this.score}`, {
fontSize: "28px",
fill: "#ffffff",
align: "center",
stroke: "#000",
strokeThickness: 3
}).setOrigin(0.5);
}
}

--- game.js ---

//game.js
import { GameScene } from './gamescene.js';
import { languageData } from './language.js';
import { setupHandsTracking, setupWebcamToggle } from './handsModule.js';
const canvasElement = document.getElementById("canvas");
const toggleBtn = document.getElementById("Nut\_Webcam");
const hands = setupHandsTracking(canvasElement, toggleBtn);
setupWebcamToggle(toggleBtn, hands);
let game;
let speedDown = 50;
let basketSpeed = 200;
function updateTexts() {
const lang = localStorage.getItem("language") || "vi";
const texts = languageData[lang];
document.getElementById("gameTitle").innerText = texts.title;
document.getElementById("startGameBtn").innerText = texts.startGame;
document.getElementById("settingsBtn").innerText = texts.settings;
document.getElementById("fallSpeedLabel").childNodes[0].textContent = texts.fallSpeed + " ";
document.getElementById("basketSpeedLabel").childNodes[0].textContent = texts.basketSpeed + " ";
document.getElementById("controlTypeTitle").innerText = texts.controlType;
document.getElementById("keyboardLabel").innerText = texts.keyboard;
document.getElementById("handtrackingLabel").innerText = texts.handtracking;
document.getElementById("languageSelectTitle").innerText = texts.languageSelect;
document.getElementById("saveSettingsBtn").innerText = texts.save;
document.getElementById("backToMenuBtn").innerText = texts.back;
document.getElementById("returnToMenuBtn").innerText = texts.returnMenu;
}
document.getElementById("settingsBtn").addEventListener("click", () => {
document.getElementById("mainMenu").style.display = "none";
document.getElementById("settings").style.display = "block";
});
document.getElementById("saveSettingsBtn").addEventListener("click", () => {
speedDown = parseInt(document.getElementById("speedSlider").value);
basketSpeed = parseInt(document.getElementById("basketSpeedSlider").value);
const selectedControl = document.querySelector('input[name="controlType"]:checked').value;
const selectedLanguage = document.getElementById("languageSelect").value;
const selectedGameMode = document.querySelector('input[name="gameMode"]:checked').value;
localStorage.setItem("gameMode", selectedGameMode);
localStorage.setItem("controlType", selectedControl);
localStorage.setItem("fallSpeed", speedDown);
localStorage.setItem("basketSpeed", basketSpeed);
localStorage.setItem("language", selectedLanguage);
updateTexts();
alert("Cài đặt đã được lưu!");
});
document.getElementById("backToMenuBtn").addEventListener("click", () => {
document.getElementById("settings").style.display = "none";
document.getElementById("mainMenu").style.display = "block";
});
document.getElementById("startGameBtn").addEventListener("click", () => {
document.getElementById("mainMenu").style.display = "none";
document.getElementById("gameScreen").style.display = "block";
if (game) game.destroy(true);
game = new Phaser.Game({
type: Phaser.WEBGL,
width: 500,
height: 500,
canvas: document.querySelector("#gameCanvas"),
physics: { default: "arcade", arcade: { gravity: { y: 0 } } },
scene: [GameScene],
});
});
document.getElementById("returnToMenuBtn").addEventListener("click", () => {
document.getElementById("gameScreen").style.display = "none";
document.getElementById("mainMenu").style.display = "block";
if (game) game.destroy(true);
});
updateTexts();