《数字找茬》软件V1.0

cc.Class({

extends: cc.Component,

properties: {

digital\_Card: {

default: null,

type: cc.Prefab

},

add\_Score: {

default: null,

type: cc.Prefab

},

Add\_ScoreAnimation: {

default: null,

type: cc.Node

},

Current\_Digital: {

default: null,

type: cc.Node

},

Ui\_Score: {

default: null,

type: cc.Node

},

CardMoveTier: {

default: null,

type: cc.Node

},

Ui\_Time: {

default: null,

type: cc.Node

},

Audio\_Correct: {

default: null,

type: cc.AudioSource

},

Audio\_Error: {

default: null,

type: cc.AudioSource

},

Audio\_Flip: {

default: null,

type: cc.AudioSource

},

Audio\_AddScore: {

default: null,

type: cc.AudioSource

},

Audio\_Init: {

default: null,

type: cc.AudioSource

},

Audio\_Finish: {

default: null,

type: cc.AudioSource

},

GameOver\_01: {

default: null,

type: cc.Node

},

StartNode: {

default: null,

type: cc.Node

},

},

pSet: null,

maxNumber: null,

modeFlag: null,

onLoad() {

this.initStart();

},

initStart: function () {

let start = this.StartNode.getComponent(cc.Animation);

start.play('StartNode\_Appear');

},

initGame\_01: function () {

this.modeFlag=1; this.Current\_Digital.children[0].getComponent(cc.Label).string = 1;

Let current = this.Current\_Digital.getComponent(cc.Animation);

current.play('CurrentAppear');

this.Ui\_Score.children[2].active = false;

this.Ui\_Score.children[1].active = true;

this.Ui\_Score.children[1].getComponent(cc.Label).string = 0;

let hint = this.Ui\_Score.getComponent(cc.Animation);

hint.play('scoreAppear');

this.Ui\_Time.children[0].getComponent(cc.Label).string = 30;

let time = this.Ui\_Time.getComponent(cc.Animation);

time.play('timeAppear');

this.maxNumber = 20;

this.squareArrangement(4, 5, 4);

this.countDown();

},

initGame\_02: function () {

this.modeFlag = 2;

this.Current\_Digital.children[0].getComponent(cc.Label).string = 1;

let current = this.Current\_Digital.getComponent(cc.Animation);

current.play('CurrentAppear');

this.Ui\_Score.children[1].active = false;

this.Ui\_Score.children[2].active = true;

this.Ui\_Score.children[2].getComponent(cc.Label).string = 0;

let hint = this.Ui\_Score.getComponent(cc.Animation);

hint.play('scoreAppear');

this.Ui\_Time.children[0].getComponent(cc.Label).string = 5;

let time = this.Ui\_Time.getComponent(cc.Animation);

time.play('timeAppear');

this.squareArrangement(4, 5, 4);

this.countDown();

},

squareArrangement: function (row, col, spacing) {

let card = cc.instantiate(this.digital\_Card);

let beginX = (this.node.width - (row - 1) \* (spacing + card.width)) / 2;

let beginY = (this.node.height - (col - 1) \* (spacing + card.height)) / 2;

card.destroy();

card.removeFromParent();

this.pSet = [];

for (let i = 0; i <= row - 1; i++) {

let array = [];

for (let j = 0; j <= col - 1; j++) {

let position = cc.v2(beginX + i \* (card.width + spacing), beginY + j \* (card.height + spacing));

array.push(position);

}

this.pSet.push(array);

}

for (let k = 0; k <= this.pSet.length - 1; k++) {

for (let l = 0; l <= this.pSet[k].length - 1; l++) {

let node = this.node;

let card = cc.instantiate(this.digital\_Card);

node.addChild(card, 0, 'card');

card.setScale(0.1);

if (l % 2 === 0) {

card.setPosition(this.pSet[k][l].x - 300, this.pSet[k][l].y - 100);

} else {

card.setPosition(this.pSet[k][l].x + 300, this.pSet[k][l].y + 100);

}

this.flopAction(card, this.pSet[k][l]);

}

}

if (this.modeFlag === 1) {

this.cardNumberInit\_01();

} else {

this.cardNumberInit\_02(row, col);

}

},

cardNumberInit\_01: function () {

let numberArr = [];

let count = this.maxNumber;

while (count > 0) {

numberArr.push(count);

count--;

}

let outOfOrder = [];

while (numberArr.length > 0) {

let index = Math.floor(Math.random() \* (numberArr.length));

outOfOrder.push(numberArr[index]);

numberArr.splice(index, 1);

}

for (let m = 0; m <= this.node.children.length - 1; m++) {

this.node.children[m].children[1].getComponent(cc.Label).string = outOfOrder[m];

}

},

cardNumberInit\_02: function (col, row) {

let number\_50 = [];

for (let i = 1; i <= 50; i++) {

number\_50.push(i);

}

let numberArr = [];

for (let j = 0; j < col \* row; j++) {

let index = Math.floor(Math.random() \* (number\_50.length));

numberArr.push(number\_50[index]);

number\_50.splice(index, 1);

}

for (let k = 0; k <= this.node.children.length - 1; k++) {

this.node.children[k].children[1].getComponent(cc.Label).string = numberArr[k];

this.scoreFormatting(this.node.children[k].children[1]);

}

this.Current\_Digital.children[0].getComponent(cc.Label).string = numberArr[Math.floor(Math.random() \* numberArr.length)];

},

flopAction: function (card, position) {

let flop = cc.callFunc(function () {

card.children[2].active = false;

}, this);

let event = cc.callFunc(function () {

if (this.modeFlag === 1) {

this.cardClickOnTheEvent\_01(card);

} else {

this.cardClickOnTheEvent\_02(card);

}

}, this);

let action1 = cc.moveTo(0.5, position.x, position.y);

action1.easing(cc.easeCircleActionInOut());

let action2 = cc.sequence(cc.scaleTo(0.1, 0.6, 0.6), cc.scaleTo(0.1, 0, 0.6), flop, cc.scaleTo(0.31, 1, 1), event);

let set = cc.spawn(action1, action2);

card.runAction(set);

},

cardClickOnTheEvent\_01: function (node) {

node.on('touchstart', function f(event) {

let numberArr = [];

for (let i = 0; i <= this.node.children.length - 1; i++) {

numberArr.push(this.node.children[i].children[1].getComponent(cc.Label).string);

}

numberArr.sort(function (a, b) {

return a - b;

});

if (node.children[1].getComponent(cc.Label).string === numberArr[0]) {

this.Audio\_Correct.play('Audio\_correct');

this.cardMovementAnimation\_01(node);

this.clickOnOk\_01();

} else {

this.Audio\_Error.play('Audio\_error');

this.clickTheError(node);

}

}, this);

},

cardClickOnTheEvent\_02: function (node) {

node.on('touchstart', function f(event) {

if(node.children[1].getComponent(cc.Label).string=== this.Current\_Digital.children[0].getComponent(cc.Label).string) {

this.Audio\_Correct.play('Audio\_correct');

this.unscheduleAllCallbacks();

this.cardMovementAnimation\_02(node);

} else {

this.Audio\_Error.play('Audio\_error');

this.clickTheError(node);

}

}, this);

},

clickTheError: function (node) {

let offEvent = cc.callFunc(function () {

if (this.modeFlag === 1) {

node.off('touchstart', this.cardClickOnTheEvent\_01(node), this);

} else {

node.off('touchstart', this.cardClickOnTheEvent\_02(node), this);

}

}, this);

let onEvent = cc.callFunc(function () {

if (this.modeFlag === 1) {

this.cardClickOnTheEvent\_01(node);

} else {

this.cardClickOnTheEvent\_02(node);

}

}, this);

let originalPos = node.position;

let act1 = cc.moveTo(0.1, originalPos.x - 10, originalPos.y);

let act2 = cc.moveTo(0.1, originalPos.x + 10, originalPos.y);

let act3 = cc.moveTo(0.1, originalPos.x, originalPos.y);

let actSet = cc.sequence(offEvent, act1, act2, act1, act3, onEvent);

node.runAction(actSet);

let hintAct1 = cc.moveTo(0.1, 230 - 8, 440);

let hintAct2 = cc.moveTo(0.1, 230 + 8, 440);

let hintAct3 = cc.moveTo(0.1, 230, 440);

let hintActSet = cc.sequence(hintAct1, hintAct2, hintAct3); this.Current\_Digital.runAction(hintActSet);

},

clickOnOk\_01: function () {

this.Ui\_Score.children[1].getComponent(cc.Label).string = Math.floor(Math.random() \* 5) + 2;

let hintAct1 = cc.moveTo(0.1, 0, 437 - 8);

let hintAct2 = cc.moveTo(0.1, 0, 437 + 8);

let hintAct3 = cc.moveTo(0.1, 0, 437);

let hintActSet = cc.sequence(hintAct1, hintAct2, hintAct3); this.Ui\_Score.runAction(hintActSet);

},

clickOnOk\_02: function (moveNode) {

this.Audio\_Flip.play('Audio\_flip');

let resetHintNumber = cc.callFunc(function () {

this.resetTime();

}, this);

this.Ui\_Score.children[2].getComponent(cc.Label).string+= moveNode.children[1].getComponent(cc.Label).string;

this.scoreFormatting(this.Ui\_Score.children[2]);

let hintAct1 = cc.moveTo(0.2, 0, 437 - 10);

let hintAct2 = cc.moveTo(0.1, 0, 437 + 10);

let hintAct3 = cc.moveTo(0.1, 0, 437);

let hintActSet = cc.sequence(hintAct1, hintAct2, hintAct3, resetHintNumber);

this.Ui\_Score.runAction(hintActSet);

let addScore = cc.instantiate(this.add\_Score);

addScore.getComponent(cc.Label).string = '/' + moveNode.children[1].getComponent(cc.Label).string;

this.Add\_ScoreAnimation.addChild(addScore);

addScore.setPosition(0, 30);

let destroy = cc.callFunc(function () {

addScore.destroy();

}, this);

let move = cc.moveTo(0.3, 0, 110);

move.easing(cc.easeIn(0.6));

let fade = cc.fadeTo(0.2, 0);

let set = cc.sequence(move, fade, destroy);

addScore.runAction(set);

},

cardMovementAnimation\_01: function (node) {

let position = node.position;

let moveNode = cc.instantiate(this.digital\_Card);

moveNode.children[1].getComponent(cc.Label).string = node.children[1].getComponent(cc.Label).string;

moveNode.setPosition(position);

this.CardMoveTier.addChild(moveNode, 0, 'star');

moveNode.children[2].active = false;

node.removeFromParent();

let end = cc.callFunc(function () {

moveNode.destroy();

moveNode.removeFromParent();

}, this);

let labelDisappear = cc.callFunc(function () {

this.Audio\_AddScore.play('Audio\_addScore');

moveNode.children[1].active = false;

this.Current\_Digital.children[0].getComponent(cc.Label).string = moveNode.children[1].getComponent(cc.Label).string + 1;

}, this);

let move = cc.callFunc(function () {

let targetPosition;

targetPosition = this.Current\_Digital.getPosition();

let moveToPos = cc.v2(Math.abs(this.node.position.x) + targetPosition.x, Math.abs(Number(this.node.position.y)) + targetPosition.y);

let scatter = cc.scaleTo(0.1, 0.85);

let fadeOut1 = cc.fadeTo(0.5, 100);

let fadeOut2 = cc.fadeTo(0.3, 0);

let cardMove = cc.moveTo(0.5, moveToPos.x, moveToPos.y);

cardMove.easing(cc.easeCircleActionInOut());

let moveTarget = cc.sequence(cardMove, labelDisappear);

let action1 = cc.scaleTo(0.05, 0, 1);

let action2 = cc.scaleTo(0.1, 1, 1);

let action3 = cc.scaleTo(0.1, 0, 1);

let action4 = cc.scaleTo(0.15, 0.8, 0.8);

let set = cc.sequence(action1, action2, action3, action4);

let rotateMove = cc.spawn(moveTarget, set);

let moveAction = cc.sequence(rotateMove, cc.scaleTo(0.05, 0.75), scatter, fadeOut2, end);

let combinationAction = cc.spawn(cc.scaleTo(0.3, 0.75), fadeOut1, moveAction);

moveNode.runAction(combinationAction);

}, this);

let generateANewCard = cc.callFunc(function () {

if (this.Ui\_Time.children[0].getComponent(cc.Label).string > 1) {

if (this.modeFlag === 1) {

this.generateANewCard(position);

} else {

this.generateANewCard(position);

this.scheduleOnce(function () {

this.Audio\_AddScore.play('Audio\_addScore');

}, 0.1);

}

}

}, this);

let pressAction = cc.sequence(cc.scaleTo(0.12, 0.85), cc.scaleTo(0.05, 1));

let pressTheAnimation = cc.sequence(pressAction, move, generateANewCard);

moveNode.runAction(pressTheAnimation);

},

cardMovementAnimation\_02: function (node) {

let position = node.position;

let moveNode = cc.instantiate(this.digital\_Card);

moveNode.children[1].getComponent(cc.Label).string = node.children[1].getComponent(cc.Label).string;

moveNode.setPosition(position);

this.CardMoveTier.addChild(moveNode, 0, 'star');

moveNode.children[2].active = false;

node.removeFromParent();

let end = cc.callFunc(function () {

moveNode.destroy();

moveNode.removeFromParent();

}, this);

let labelDisappear = cc.callFunc(function () {

moveNode.children[2].active = false;

this.clickOnOk\_02(moveNode);

}, this);

let move = cc.callFunc(function () {

let targetPosition = this.Ui\_Score.getPosition();

let moveToPos = cc.v2(Math.abs(this.node.position.x) + targetPosition.x, Math.abs(Number(this.node.position.y)) + targetPosition.y);

let cardMove = cc.moveTo(0.5, moveToPos.x, moveToPos.y + 20);

cardMove.easing(cc.easeCircleActionInOut());

let moveTarget = cc.sequence(cardMove, labelDisappear);

let action1 = cc.scaleTo(0.05, 0, 1);

let action2 = cc.scaleTo(0.1, 1, 1);

let action3 = cc.scaleTo(0.1, 0, 1);

let action4 = cc.scaleTo(0.1, 0.7, 0.7);

let set = cc.sequence(action1, action2, action3, action4);

let rotateMove = cc.spawn(moveTarget, set);

let moveAction = cc.sequence(rotateMove, end);

moveNode.runAction(moveAction);

let fadeOut1 = cc.fadeTo(0.2, 255);

let fadeOut2 = cc.fadeTo(0.2, 0);

let children\_0\_set1 = cc.sequence(fadeOut1, fadeOut2);

moveNode.children[0].runAction(children\_0\_set1);

let fadeOut3 = cc.fadeTo(0.2, 255);

let fadeOut4 = cc.fadeTo(0.1, 0);

let children\_1\_set1 = cc.sequence(fadeOut3, fadeOut4);

moveNode.children[1].runAction(children\_1\_set1);

}, this);

let generateANewCard = cc.callFunc(function () {

if (this.Ui\_Time.children[0].getComponent(cc.Label).string > 0) {

this.scheduleOnce(function () {

this.Audio\_AddScore.play('Audio\_addScore');

}, 0.35);

this.generateANewCard(position);

}

}, this);

let pressAction = cc.sequence(cc.scaleTo(0.12, 0.85), cc.scaleTo(0.05, 1));

let pressTheAnimation = cc.sequence(pressAction, move, generateANewCard);

moveNode.runAction(pressTheAnimation);

},

generateANewCard: function (position) {

let numberArr = [];

for (let k = 0; k < this.node.children.length - 1; k++) {

numberArr.push(this.node.children[k].children[1].getComponent(cc.Label).string);

}

numberArr.sort(function (a, b) {

return a - b;

});

let newCard = cc.instantiate(this.digital\_Card);

if (this.modeFlag === 1) {

this.maxNumber++;

newCard.children[1].getComponent(cc.Label).string = this.maxNumber;

this.node.addChild(newCard, 0, 'card');

} else {

let randomNumber = Math.floor(Math.random() \* 50) + 1;

this.node.addChild(newCard, 0, 'card');

newCard.children[1].getComponent(cc.Label).string = numberArr[numberArr.length - 1] + randomNumber;

this.scoreFormatting(newCard.children[1]);

}

if (position.x > this.node.width / 2 && position.y > this.node.height / 2) {

newCard.setPosition(650, 950);

} else if (position.x < this.node.width / 2 && position.y < this.node.height / 2) {

newCard.setPosition(0, 0);

} else if (position.x > this.node.width / 2 && position.y < this.node.height / 2) {

newCard.setPosition(650, 0);

} else if (position.x < this.node.width / 2 && position.y > this.node.height / 2) {

newCard.setPosition(0, 950);

}

this.flopAction(newCard, position, this.modeFlag);

},

scoreFormatting: function (node) {

let length = node.getComponent(cc.Label).string.toString().length;

let label = node.getComponent(cc.Label);

switch (length) {

case 1:

label.fontSize = 42;

break;

case 2:

label.fontSize = 40;

break;

case 3:

label.fontSize = 38;

break;

case 4:

label.fontSize = 35;

break;

case 5:

label.fontSize = 33;

break;

case 6:

label.fontSize = 30;

break;

case 7:

label.fontSize = 28;

break;

case 8:

label.fontSize = 25;

break;

default:

label.fontSize = 23;

}

},

resetTime: function () {

this.Audio\_Flip.play('Audio\_flip');//翻牌声音

let resetTime = cc.callFunc(function () {

this.Ui\_Time.children[0].getComponent(cc.Label).string = 5;

}, this);

let resetHintNumber = cc.callFunc(function () {

this.resetHintNumber();

}, this);

let scale\_01 = cc.scaleTo(0.05, 0, 1);

let scale\_02 = cc.scaleTo(0.1, 1, 1);

let set\_01 = cc.sequence(scale\_01, resetTime, scale\_02, resetHintNumber);

let move\_01 = cc.moveTo(0.1, -230, 440 + 10);

let move\_02 = cc.moveTo(0.1, -230, 440);

let set\_02 = cc.sequence(move\_01, move\_02);

let set = cc.spawn(set\_01, set\_02);

this.Ui\_Time.runAction(set);

},

resetHintNumber: function () {

let resetNumber = cc.callFunc(function () {

let index = Math.floor(Math.random() \* this.node.children.length);

this.Current\_Digital.children[0].getComponent(cc.Label).string = this.node.children[index].children[1].getComponent(cc.Label).string;

}, this);

let continueGame = cc.callFunc(function () {

this.countDown();

}, this);

let scale\_01 = cc.scaleTo(0.05, 0, 1);

let scale\_02 = cc.scaleTo(0.1, 1, 1);

let set\_01 = cc.sequence(scale\_01, resetNumber, scale\_02, continueGame);

let move\_01 = cc.moveTo(0.1, 230, 440 + 10);

let move\_02 = cc.moveTo(0.1, 230, 440);

let set\_02 = cc.sequence(move\_01, move\_02);

let set = cc.spawn(set\_01, set\_02);

this.Current\_Digital.runAction(set);

},

countDown: function () {

this.callback = function () {

this.Ui\_Time.children[0].getComponent(cc.Label).string--;

if (this.Ui\_Time.children[0].getComponent(cc.Label).string === 0) {

this.cardVanishing();

this.gameOverAnimation();

this.unschedule(this.callback);

}

};

this.schedule(this.callback, 1);

},

cardVanishing: function () {

for (let i = 0; i < this.node.children.length; i++) {

this.node.children[i].off('touchstart', this.cardClickOnTheEvent\_01(this.node.children[i]), this);

if(this.node.children[i].children[1].getComponent(cc.Label).string === this.Current\_Digital.children[0].getComponent(cc.Label).string) {

let move = cc.callFunc(function () {

let destroy = cc.callFunc(function () {

this.node.removeAllChildren();

}, this);

if (this.node.children[i].position.x > this.node.width / 2) {

let nodeMove\_01 = cc.moveBy(0.2, 800, 0);

nodeMove\_01.easing(cc.easeCircleActionInOut());

let nodeMoveSet\_01 = cc.sequence(nodeMove\_01, destroy);

this.node.children[i].runAction(nodeMoveSet\_01);

} else {

let nodeMove\_02 = cc.moveBy(0.2, -800, 0);

nodeMove\_02.easing(cc.easeCircleActionInOut());

let nodeMoveSet\_02 = cc.sequence(nodeMove\_02, destroy);

this.node.children[i].runAction(nodeMoveSet\_02);

}

}, this);

let audio = cc.callFunc(function () {

this.Audio\_Error.play('Audio\_error');

}, this);

let targetStop\_01 = cc.scaleTo(0.6, 1);

let targetStop\_02 = cc.scaleTo(0.4, 1);

let targetMove\_01 = cc.moveBy(0.1, 25, 0);

let targetMove\_02 = cc.moveBy(0.1, -50, 0);

let targetMove\_03 = cc.moveBy(0.1, 25, 0);

let targetMoveSet = cc.sequence(targetStop\_01, audio, targetMove\_01, targetMove\_02, targetMove\_03, targetStop\_02, move);

this.node.children[i].runAction(targetMoveSet);

} else {

let fade\_01 = cc.fadeTo(0.3, 0);

let scale\_01 = cc.scaleTo(0.1, 1.05);

let scale\_02 = cc.scaleTo(0.3, 0.8);

let set\_01 = cc.sequence(scale\_01, scale\_02);

let set\_02 = cc.spawn(set\_01, fade\_01);

this.node.children[i].runAction(set\_02);

}

}

},

gameOverAnimation: function () {

this.scheduleOnce(function () {

let uiTime = this.Ui\_Time.getComponent(cc.Animation);

uiTime.play('timeMove');

}, 1.2);

this.scheduleOnce(function () {

if (this.modeFlag === 1) {

let action = cc.moveTo(0.2, 0, 800);

this.Ui\_Score.runAction(action);

} else {

let digitalAnimation = this.Current\_Digital.getComponent(cc.Animation);

digitalAnimation.play('CurrentSwitcher\_02');

}

}, 1.3);

this.scheduleOnce(function () {

this.Audio\_Finish.play('Audio\_finish');

if (this.modeFlag === 1) {

let digitalAnimation = this.Current\_Digital.getComponent(cc.Animation);

digitalAnimation.play('CurrentMove');

} else {

let score = this.Ui\_Score.getComponent(cc.Animation);

score.play('scoreShow');

}

this.GameOver\_01.active = true;

let GameOver\_01 = this.GameOver\_01.getComponent(cc.Animation);

GameOver\_01.play('GameOver\_01Animation');

}, 1.4);

},

});

cc.Class({

extends: cc.Component,

properties: {

GameOver\_01: {

default: null,

type: cc.Node

},

Button\_again: {

default: null,

type: cc.Button

},

Button\_menu: {

default: null,

type: cc.Button

},

Current\_Digital: {

default: null,

type: cc.Node

},

Audio\_Button: {

default: null,

type: cc.AudioSource

},

GameLayout: {

default: null,

type: cc.Node

},

Ui\_Score: {

default: null,

type: cc.Node

},

StartNode: {

default: null,

type: cc.Node

},

},

againButtonEvent: function () {

this.interfaceSwitch();

this.scheduleOnce(function () {

this.GameLayout.removeAllChildren();

let GameScript = this.GameLayout.getComponent('GameScript');

if (GameScript.modeFlag === 1) {

GameScript.initGame\_01();

} else {

GameScript.initGame\_02();

}

}, 0.5);

},

menuButtonEvent: function () {

this.interfaceSwitch();

this.GameLayout.removeAllChildren();

let startNode = this.StartNode.getComponent(cc.Animation);

startNode.play('StartNode\_Appear');

},

interfaceSwitch: function () {

this.Audio\_Button.play('Audio\_button');

let GameOver = this.node.getComponent(cc.Animation);

GameOver.play('GameOver\_01Switcher');

let GameScript = this.GameLayout.getComponent('GameScript');

if (GameScript.modeFlag === 1) {

let current = this.Current\_Digital.getComponent(cc.Animation);

current.play('CurrentSwitcher');

} else {

let score = this.Ui\_Score.getComponent(cc.Animation);

score.play('scoreSwitcher');

}

},

gameOverAwait: function () {

this.node.active = true;

let GameOver = this.node.getComponent(cc.Animation);

GameOver.play('GameOver\_await');

},

});

cc.Class({

extends: cc.Component,

properties: {

Button\_Start\_01: {

default: null,

type: cc.Button

},

Button\_Start\_02: {

default: null,

type: cc.Button

},

GameLayout: {

default: null,

type: cc.Node

},

GameOver\_01: {

default: null,

type: cc.Node

},

Audio\_Button: {

default: null,

type: cc.AudioSource

},

},

Button\_Start\_01Clink: function () {

this.Audio\_Button.play('Audio\_button');

let start = this.node.getComponent(cc.Animation);

start.play('StartNode\_Move');

this.GameLayout.removeAllChildren();

this.scheduleOnce(function () {

this.GameLayout.removeAllChildren();

this.GameLayout.getComponent('GameScript').initGame\_01();

}, 0.5);

},

Button\_Start\_02Clink: function () {

this.Audio\_Button.play('Audio\_button');

let start = this.node.getComponent(cc.Animation);

start.play('StartNode\_Move');

this.GameLayout.removeAllChildren();

this.scheduleOnce(function () {

this.GameLayout.removeAllChildren();

this.GameLayout.getComponent('GameScript').initGame\_02();

}, 0.5);

},

});

(--结束(完)--)