cc.PRIORITY\_NON\_SYSTEM = cc.PRIORITY\_SYSTEM + 1;

cc.ListEntry = function (prev, next, callback, target, priority, paused, markedForDeletion) {

this.prev = prev;

this.next = next;

this.callback = callback;

this.target = target;

this.priority = priority;

this.paused = paused;

this.markedForDeletion = markedForDeletion;

};

cc.HashUpdateEntry = function (list, entry, target, callback, hh) {

this.list = list;

this.entry = entry;

this.target = target;

this.callback = callback;

this.hh = hh;

};

cc.HashTimerEntry = cc.hashSelectorEntry = function (timers, target, timerIndex, currentTimer, currentTimerSalvaged, paused, hh) {

var \_t = this;

\_t.timers = timers;

\_t.target = target;

\_t.timerIndex = timerIndex;

\_t.currentTimer = currentTimer;

\_t.currentTimerSalvaged = currentTimerSalvaged;

\_t.paused = paused;

\_t.hh = hh;

};

cc.Timer = cc.Class.extend({

\_scheduler: null,

\_elapsed: 0.0,

\_runForever: false,

\_useDelay: false,

\_timesExecuted: 0,

\_repeat: 0,

\_delay: 0,

\_interval: 0.0,

getInterval: function () {

return this.\_interval;

},

setInterval: function (interval) {

this.\_interval = interval;

},

setupTimerWithInterval: function (seconds, repeat, delay) {

this.\_elapsed = -1;

this.\_interval = seconds;

this.\_delay = delay;

this.\_useDelay = (this.\_delay > 0);

this.\_repeat = repeat;

this.\_runForever = (this.\_repeat === cc.REPEAT\_FOREVER);

},

trigger: function () {

return 0;

},

cancel: function () {

return 0;

},

ctor: function () {

this.\_scheduler = null;

this.\_elapsed = -1;

this.\_runForever = false;

this.\_useDelay = false;

this.\_timesExecuted = 0;

this.\_repeat = 0;

this.\_delay = 0;

this.\_interval = 0;

},

update: function (dt) {

if (this.\_elapsed === -1) {

this.\_elapsed = 0;

this.\_timesExecuted = 0;

} else {

this.\_elapsed += dt;

if (this.\_runForever && !this.\_useDelay) {

if (this.\_elapsed >= this.\_interval) {

this.trigger();

this.\_elapsed = 0;

}

} else {

if (this.\_useDelay) {

if (this.\_elapsed >= this.\_delay) {

this.trigger();

this.\_elapsed -= this.\_delay;

this.\_timesExecuted += 1;

this.\_useDelay = false;

}

} else {

if (this.\_elapsed >= this.\_interval) {

this.trigger();

this.\_elapsed = 0;

this.\_timesExecuted += 1;

}

}

if (!this.\_runForever && this.\_timesExecuted > this.\_repeat) this.cancel();

}

}

}

});

cc.TimerTargetSelector = cc.Timer.extend({

\_target: null,

\_selector: null,

ctor: function () {

this.\_target = null;

this.\_selector = null;

},

initWithSelector: function (scheduler, selector, target, seconds, repeat, delay) {

this.\_scheduler = scheduler;

this.\_target = target;

this.\_selector = selector;

this.setupTimerWithInterval(seconds, repeat, delay);

return true;

},

getSelector: function () {

return this.\_selector;

},

trigger: function () {

if (this.\_target && this.\_selector) {

this.\_target.call(this.\_selector, this.\_elapsed);

}

},

cancel: function () {

this.\_scheduler.unschedule(this.\_selector, this.\_target);

}

});

cc.TimerTargetCallback = cc.Timer.extend({

\_target: null,

\_callback: null,

\_key: null,

ctor: function () {

this.\_target = null;

this.\_callback = null;

},

initWithCallback: function (scheduler, callback, target, key, seconds, repeat, delay) {

this.\_scheduler = scheduler;

this.\_target = target;

this.\_callback = callback;

this.\_key = key;

this.setupTimerWithInterval(seconds, repeat, delay);

return true;

},

getCallback: function () {

return this.\_callback;

},

getKey: function () {

return this.\_key;

},

trigger: function () {

if (this.\_callback) this.\_callback.call(this.\_target, this.\_elapsed);

},

cancel: function () {

this.\_scheduler.unschedule(this.\_callback, this.\_target);

}

});

cc.Scheduler = cc.Class.extend({

\_timeScale: 1.0,

\_updatesNegList: null,

\_updates0List: null,

\_updatesPosList: null,

\_hashForTimers: null,

\_arrayForTimers: null,

\_hashForUpdates: null,

\_currentTarget: null,

\_currentTargetSalvaged: false,

\_updateHashLocked: false,

ctor: function () {

this.\_timeScale = 1.0;

this.\_updatesNegList = [];

this.\_updates0List = [];

this.\_updatesPosList = [];

this.\_hashForUpdates = {};

this.\_hashForTimers = {};

this.\_currentTarget = null;

this.\_currentTargetSalvaged = false;

this.\_updateHashLocked = false;

this.\_arrayForTimers = [];

},

\_schedulePerFrame: function (callback, target, priority, paused) {

var hashElement = this.\_hashForUpdates[target.\_\_instanceId];

if (hashElement && hashElement.entry) {

if (hashElement.entry.priority !== priority) {

if (this.\_updateHashLocked) {

cc.log("warning: you CANNOT change update priority in scheduled function");

hashElement.entry.markedForDeletion = false;

hashElement.entry.paused = paused;

return;

} else {

this.unscheduleUpdate(target);

}

} else {

hashElement.entry.markedForDeletion = false;

hashElement.entry.paused = paused;

return;

}

}

if (priority === 0) {

this.\_appendIn(this.\_updates0List, callback, target, paused);

} else if (priority < 0) {

this.\_priorityIn(this.\_updatesNegList, callback, target, priority, paused);

} else {

this.\_priorityIn(this.\_updatesPosList, callback, target, priority, paused);

}

},

\_removeHashElement: function (element) {

delete this.\_hashForTimers[element.target.\_\_instanceId];

cc.arrayRemoveObject(this.\_arrayForTimers, element);

element.Timer = null;

element.target = null;

element = null;

},

\_removeUpdateFromHash: function (entry) {

var self = this,

element = self.\_hashForUpdates[entry.target.\_\_instanceId];

if (element) {

cc.arrayRemoveObject(element.list, element.entry);

delete self.\_hashForUpdates[element.target.\_\_instanceId];

element.entry = null;

element.target = null;

}

},

\_priorityIn: function (ppList, callback, target, priority, paused) {

var self = this,

listElement = new cc.ListEntry(null, null, callback, target, priority, paused, false);

if (!ppList) {

ppList = [];

ppList.push(listElement);

} else {

var index2Insert = ppList.length - 1;

for (var i = 0; i <= index2Insert; i++) {

if (priority < ppList[i].priority) {

index2Insert = i;

break;

}

}

ppList.splice(i, 0, listElement);

}

self.\_hashForUpdates[target.\_\_instanceId] = new cc.HashUpdateEntry(ppList, listElement, target, null);

return ppList;

},

\_appendIn: function (ppList, callback, target, paused) {

var self = this,

listElement = new cc.ListEntry(null, null, callback, target, 0, paused, false);

ppList.push(listElement);

self.\_hashForUpdates[target.\_\_instanceId] = new cc.HashUpdateEntry(ppList, listElement, target, null, null);

},

setTimeScale: function (timeScale) {

this.\_timeScale = timeScale;

},

getTimeScale: function () {

return this.\_timeScale;

},

update: function (dt) {

this.\_updateHashLocked = true;

if (this.\_timeScale !== 1) dt \*= this.\_timeScale;

var i, list, len, entry;

for (i = 0, list = this.\_updatesNegList, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

for (i = 0, list = this.\_updates0List, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

for (i = 0, list = this.\_updatesPosList, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

var elt, arr = this.\_arrayForTimers;

for (i = 0; i < arr.length; i++) {

elt = arr[i];

this.\_currentTarget = elt;

this.\_currentTargetSalvaged = false;

if (!elt.paused) {

for (elt.timerIndex = 0; elt.timerIndex < elt.timers.length; ++(elt.timerIndex)) {

elt.currentTimer = elt.timers[elt.timerIndex];

elt.currentTimerSalvaged = false;

elt.currentTimer.update(dt);

elt.currentTimer = null;

}

}

if (this.\_currentTargetSalvaged && this.\_currentTarget.timers.length === 0) this.\_removeHashElement(this.\_currentTarget);

}

for (i = 0, list = this.\_updatesNegList; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

for (i = 0, list = this.\_updates0List; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

for (i = 0, list = this.\_updatesPosList; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

this.\_updateHashLocked = false;

this.\_currentTarget = null;

},

scheduleCallbackForTarget: function (target, callback\_fn, interval, repeat, delay, paused) {

this.schedule(callback\_fn, target, interval, repeat, delay, paused, target.\_\_instanceId + "");

},

schedule: function (callback, target, interval, repeat, delay, paused, key) {

var isSelector = false;

if (typeof callback !== "function") {

var selector = callback;

isSelector = true;

}

if (isSelector === false) {

if (arguments.length === 4 || arguments.length === 5) {

key = delay;

paused = repeat;

delay = 0;

repeat = cc.REPEAT\_FOREVER;

}

} else {

if (arguments.length === 4) {

paused = repeat;

repeat = cc.REPEAT\_FOREVER;

delay = 0;

}

}

if (key === undefined) {

key = target.\_\_instanceId + "";

}

cc.assert(target, cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget\_3);

var element = this.\_hashForTimers[target.\_\_instanceId];

if (!element) {

element = new cc.HashTimerEntry(null, target, 0, null, null, paused, null);

this.\_arrayForTimers.push(element);

this.\_hashForTimers[target.\_\_instanceId] = element;

} else {

cc.assert(element.paused === paused, "");

}

var timer, i;

if (element.timers == null) {

element.timers = [];

} else if (isSelector === false) {

for (i = 0; i < element.timers.length; i++) {

timer = element.timers[i];

if (callback === timer.\_callback) {

cc.log(cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget, timer.getInterval().toFixed(4), interval.toFixed(4));

timer.\_interval = interval;

return;

}

}

} else {

for (i = 0; i < element.timers.length; ++i) {

timer = element.timers[i];

if (timer && selector === timer.getSelector()) {

cc.log("CCScheduler#scheduleSelector. Selector already scheduled. Updating interval from: %.4f to %.4f", timer.getInterval(), interval);

timer.setInterval(interval);

return;

}

}

}

if (isSelector === false) {

timer = new cc.TimerTargetCallback();

timer.initWithCallback(this, callback, target, key, interval, repeat, delay);

element.timers.push(timer);

} else {

timer = new cc.TimerTargetSelector();

timer.initWithSelector(this, selector, target, interval, repeat, delay);

element.timers.push(timer);

}

},

scheduleUpdate: function (target, priority, paused) {

this.\_schedulePerFrame(function (dt) {

target.update(dt);

}, target, priority, paused);

},

\_getUnscheduleMark: function (key, timer) {

switch (typeof key) {

case "number":

case "string":

return key === timer.getKey();

case "function":

return key === timer.\_callback;

default:

return key === timer.getSelector();

}

},

unschedule: function (key, target) {

if (!target || !key) return;

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

var timers = element.timers;

for (var i = 0, li = timers.length; i < li; i++) {

var timer = timers[i];

if (this.\_getUnscheduleMark(key, timer)) {

if ((timer === element.currentTimer) && (!element.currentTimerSalvaged)) {

element.currentTimerSalvaged = true;

}

timers.splice(i, 1);

if (element.timerIndex >= i) {

element.timerIndex--;

}

if (timers.length === 0) {

if (self.\_currentTarget === element) {

self.\_currentTargetSalvaged = true;

} else {

self.\_removeHashElement(element);

}

}

return;

}

}

}

},

unscheduleUpdate: function (target) {

if (target == null) return;

var element = this.\_hashForUpdates[target.\_\_instanceId];

if (element) {

if (this.\_updateHashLocked) {

element.entry.markedForDeletion = true;

} else {

this.\_removeUpdateFromHash(element.entry);

}

}

},

unscheduleAllForTarget: function (target) {

if (target == null) {

return;

}

var element = this.\_hashForTimers[target.\_\_instanceId];

if (element) {

if (element.timers.indexOf(element.currentTimer) > -1 && (!element.currentTimerSalvaged)) {

element.currentTimerSalvaged = true;

}

element.timers.length = 0;

if (this.\_currentTarget === element) {

this.\_currentTargetSalvaged = true;

} else {

this.\_removeHashElement(element);

}

}

this.unscheduleUpdate(target);

},

unscheduleAll: function () {

this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

unscheduleAllWithMinPriority: function (minPriority) {

var i, element, arr = this.\_arrayForTimers;

for (i = arr.length - 1; i >= 0; i--) {

element = arr[i];

this.unscheduleAllForTarget(element.target);

}

var entry;

var temp\_length = 0;

if (minPriority < 0) {

for (i = 0; i < this.\_updatesNegList.length;) {

temp\_length = this.\_updatesNegList.length;

entry = this.\_updatesNegList[i];

if (entry && entry.priority >= minPriority) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updatesNegList.length) i++;

}

}

if (minPriority <= 0) {

for (i = 0; i < this.\_updates0List.length;) {

temp\_length = this.\_updates0List.length;

entry = this.\_updates0List[i];

if (entry) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updates0List.length) i++;

}

}

for (i = 0; i < this.\_updatesPosList.length;) {

temp\_length = this.\_updatesPosList.length;

entry = this.\_updatesPosList[i];

if (entry && entry.priority >= minPriority) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updatesPosList.length) i++;

}

},

isScheduled: function (key, target) {

cc.assert(key, "Argument key must not be empty");

cc.assert(target, "Argument target must be non-nullptr");

var element = this.\_hashForUpdates[target.\_\_instanceId];

if (!element) {

return false;

}

if (element.timers == null) {

return false;

} else {

var timers = element.timers;

for (var i = 0; i < timers.length; ++i) {

var timer = timers[i];

if (key === timer.getKey()) {

return true;

}

}

return false;

}

},

pauseAllTargets: function () {

return this.pauseAllTargetsWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

pauseAllTargetsWithMinPriority: function (minPriority) {

var idsWithSelectors = [];

var self = this,

element, locArrayForTimers = self.\_arrayForTimers;

var i, li;

for (i = 0, li = locArrayForTimers.length; i < li; i++) {

element = locArrayForTimers[i];

if (element) {

element.paused = true;

idsWithSelectors.push(element.target);

}

}

var entry;

if (minPriority < 0) {

for (i = 0; i < this.\_updatesNegList.length; i++) {

entry = this.\_updatesNegList[i];

if (entry) {

if (entry.priority >= minPriority) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

}

if (minPriority <= 0) {

for (i = 0; i < this.\_updates0List.length; i++) {

entry = this.\_updates0List[i];

if (entry) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

for (i = 0; i < this.\_updatesPosList.length; i++) {

entry = this.\_updatesPosList[i];

if (entry) {

if (entry.priority >= minPriority) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

return idsWithSelectors;

},

resumeTargets: function (targetsToResume) {

if (!targetsToResume) return;

for (var i = 0; i < targetsToResume.length; i++) {

this.resumeTarget(targetsToResume[i]);

}

},

pauseTarget: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_pauseTarget);

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

element.paused = true;

}

var elementUpdate = self.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

elementUpdate.entry.paused = true;

}

},

resumeTarget: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_resumeTarget);

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

element.paused = false;

}

var elementUpdate = self.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

elementUpdate.entry.paused = false;

}

},

isTargetPaused: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_isTargetPaused);

var element = this.\_hashForTimers[target.\_\_instanceId];

if (element) {

return element.paused;

}

var elementUpdate = this.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

return elementUpdate.entry.paused;

}

return false;

},

scheduleUpdateForTarget: function (target, priority, paused) {

this.scheduleUpdate(target, priority, paused);

},

unscheduleCallbackForTarget: function (target, callback) {

this.unschedule(callback, target);

},

unscheduleUpdateForTarget: function (target) {

this.unscheduleUpdate(target);

},

unscheduleAllCallbacksForTarget: function (target) {

this.unschedule(target.\_\_instanceId + "", target);

},

unscheduleAllCallbacks: function () {

this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

unscheduleAllCallbacksWithMinPriority: function (minPriority) {

this.unscheduleAllWithMinPriority(minPriority);

}

});

cc.Scheduler.PRIORITY\_SYSTEM = (-2147483647 - 1);

CSSProperties.prototype.\_super;

CSSProperties.prototype.ctor;

CSSProperties.prototype.Inflate;

CSSProperties.prototype.decompress;

CSSProperties.prototype.DeviceOrientationEvent;

CSSProperties.prototype.DeviceMotionEvent;

CSSProperties.prototype.accelerationIncludingGravity;

CSSProperties.prototype.gamma;

CSSProperties.prototype.beta;

CSSProperties.prototype.alpha;

var gl = gl || {};

CSSProperties.prototype.gl;

CSSProperties.prototype.AudioContext;

CSSProperties.prototype.webkitAudioContext;

CSSProperties.prototype.mozAudioContext;

CSSProperties.prototype.createBufferSource;

CSSProperties.prototype.createGain;

CSSProperties.prototype.createGainNode;

CSSProperties.prototype.destination;

CSSProperties.prototype.decodeAudioData;

CSSProperties.prototype.gain;

CSSProperties.prototype.connect;

CSSProperties.prototype.playbackState;

CSSProperties.prototype.noteGrainOn;

CSSProperties.prototype.noteOn;

cc.game.addEventListener(cc.game.EVENT\_RENDERER\_INITED, function () {

if (cc.\_renderType === cc.game.RENDER\_TYPE\_CANVAS) {

var \_p = cc.Director.prototype;

\_p.getProjection = function (projection) {

return this.\_projection;

};

\_p.setProjection = function (projection) {

this.\_projection = projection;

cc.eventManager.dispatchEvent(this.\_eventProjectionChanged);

};

\_p.setDepthTest = function () {};

\_p.setClearColor = function (clearColor) {

cc.renderer.\_clearColor = clearColor;

cc.renderer.\_clearFillStyle = 'rgb(' + clearColor.r + ',' + clearColor.g + ',' + clearColor.b + ')';

};

\_p.setOpenGLView = function (openGLView) {

this.\_winSizeInPoints.width = cc.\_canvas.width;

this.\_winSizeInPoints.height = cc.\_canvas.height;

this.\_openGLView = openGLView || cc.view;

if (cc.eventManager) cc.eventManager.setEnabled(true);

};

\_p.getVisibleSize = function () {

return this.getWinSize();

};

\_p.getVisibleOrigin = function () {

return cc.p(0, 0);

};

} else {

cc.Director.\_fpsImage = new Image();

cc.Director.\_fpsImage.addEventListener("load", function () {

cc.Director.\_fpsImageLoaded = true;

});

if (cc.\_fpsImage) {

cc.Director.\_fpsImage.src = cc.\_fpsImage;

}

}

});

cc.PI2 = Math.PI \* 2;

cc.DrawingPrimitiveCanvas = cc.Class.extend({

\_cacheArray: [],

\_renderContext: null,

ctor: function (renderContext) {

this.\_renderContext = renderContext;

},

drawPoint: function (point, size) {

if (!size) {

size = 1;

}

var locScaleX = cc.view.getScaleX(),

locScaleY = cc.view.getScaleY();

var newPoint = cc.p(point.x \* locScaleX, point.y \* locScaleY);

var ctx = this.\_renderContext.getContext();

ctx.beginPath();

ctx.arc(newPoint.x, -newPoint.y, size \* locScaleX, 0, Math.PI \* 2, false);

ctx.closePath();

ctx.fill();

},

drawPoints: function (points, numberOfPoints, size) {

if (points == null) return;

if (!size) {

size = 1;

}

t.x, -(destPoint.y + destSize.height), destSize.width, destSize.height);

break;default: throw new Error("Argument must be non-nil");

break;

}

}, drawStar: function (ctx, radius, color) {

var wrapper = ctx || this.\_renderContext;

var context = wrapper.getContext();

var colorStr = "rgba(" + (0 | color.r) + "," + (0 | color.g) + "," + (0 | color.b);

wrapper.setFillStyle(colorStr + ",1)");

var subRadius = radius / 10;

context.beginPath();

context.moveTo(-radius, radius);

context.lineTo(0, subRadius);

context.lineTo(radius, radius);

context.lineTo(subRadius, 0);

context.lineTo(radius, -radius);

context.lineTo(0, -subRadius);

context.lineTo(-radius, -radius);

context.lineTo(-subRadius, 0);

context.lineTo(-radius, radius);

context.closePath();

context.fill();

var rg = context.createRadialGradient(0, 0, subRadius, 0, 0, radius);

rg.addColorStop(0, colorStr + ", 1)");

rg.addColorStop(0.3, colorStr + ", 0.8)");

rg.addColorStop(1.0, colorStr + ", 0.0)");

wrapper.setFillStyle(rg);

context.beginPath();

var startAngle\_1 = 0;

var endAngle\_1 = cc.PI2;

context.arc(0, 0, radius - subRadius, startAngle\_1, endAngle\_1, false);

context.closePath();

context.fill();

}, drawColorBall: function (ctx, radius, color) {

var wrapper = ctx || this.\_renderContext;

var context = wrapper.getContext();

radius \*= cc.view.getScaleX();

var colorStr = "rgba(" + (0 | color.r) + "," + (0 | color.g) + "," + (0 | color.b);

var subRadius = radius / 10;

var g1 = context.createRadialGradient(0, 0, subRadius, 0, 0, radius);

g1.addColorStop(0, colorStr + ", 1)");

g1.addColorStop(0.3, colorStr + ", 0.8)");

g1.addColorStop(0.6, colorStr + ", 0.4)");

g1.addColorStop(1.0, colorStr + ", 0.0)");

wrapper.setFillStyle(g1);

context.beginPath();

var startAngle\_1 = 0;

var endAngle\_1 = cc.PI2;

context.arc(0, 0, radius, startAngle\_1, endAngle\_1, false);

context.closePath();

context.fill();

}, fillText: function (strText, x, y) {

this.\_renderContext.getContext().fillText(strText, x, -y);

}, setDrawColor: function (r, g, b, a) {

this.\_renderContext.setFillStyle("rgba(" + r + "," + g + "," + b + "," + a / 255 + ")");

this.\_renderContext.setStrokeStyle("rgba(" + r + "," + g + "," + b + "," + a / 255 + ")");

}, setPointSize: function (pointSize) {}, setLineWidth: function (width) {

this.\_renderContext.getContext().lineWidth = width \* cc.view.getScaleX();

}

});

cc.PRIORITY\_NON\_SYSTEM = cc.PRIORITY\_SYSTEM + 1;

cc.ListEntry = function (prev, next, callback, target, priority, paused, markedForDeletion) {

this.prev = prev;

this.next = next;

this.callback = callback;

this.target = target;

this.priority = priority;

this.paused = paused;

this.markedForDeletion = markedForDeletion;

};

cc.HashUpdateEntry = function (list, entry, target, callback, hh) {

this.list = list;

this.entry = entry;

this.target = target;

this.callback = callback;

this.hh = hh;

};

cc.HashTimerEntry = cc.hashSelectorEntry = function (timers, target, timerIndex, currentTimer, currentTimerSalvaged, paused, hh) {

var \_t = this;

\_t.timers = timers;

\_t.target = target;

\_t.timerIndex = timerIndex;

\_t.currentTimer = currentTimer;

\_t.currentTimerSalvaged = currentTimerSalvaged;

\_t.paused = paused;

\_t.hh = hh;

};

cc.Timer = cc.Class.extend({

\_scheduler: null,

\_elapsed: 0.0,

\_runForever: false,

\_useDelay: false,

\_timesExecuted: 0,

\_repeat: 0,

\_delay: 0,

\_interval: 0.0,

getInterval: function () {

return this.\_interval;

},

setInterval: function (interval) {

this.\_interval = interval;

},

setupTimerWithInterval: function (seconds, repeat, delay) {

this.\_elapsed = -1;

this.\_interval = seconds;

this.\_delay = delay;

this.\_useDelay = (this.\_delay > 0);

this.\_repeat = repeat;

this.\_runForever = (this.\_repeat === cc.REPEAT\_FOREVER);

},

trigger: function () {

return 0;

},

cancel: function () {

return 0;

},

ctor: function () {

this.\_scheduler = null;

this.\_elapsed = -1;

this.\_runForever = false;

this.\_useDelay = false;

this.\_timesExecuted = 0;

this.\_repeat = 0;

this.\_delay = 0;

this.\_interval = 0;

},

update: function (dt) {

if (this.\_elapsed === -1) {

this.\_elapsed = 0;

this.\_timesExecuted = 0;

} else {

this.\_elapsed += dt;

if (this.\_runForever && !this.\_useDelay) {

if (this.\_elapsed >= this.\_interval) {

this.trigger();

this.\_elapsed = 0;

}

} else {

if (this.\_useDelay) {

if (this.\_elapsed >= this.\_delay) {

this.trigger();

this.\_elapsed -= this.\_delay;

this.\_timesExecuted += 1;

this.\_useDelay = false;

}

} else {

if (this.\_elapsed >= this.\_interval) {

this.trigger();

this.\_elapsed = 0;

this.\_timesExecuted += 1;

}

}

if (!this.\_runForever && this.\_timesExecuted > this.\_repeat) this.cancel();

}

}

}

});

cc.TimerTargetSelector = cc.Timer.extend({

\_target: null,

\_selector: null,

ctor: function () {

this.\_target = null;

this.\_selector = null;

},

initWithSelector: function (scheduler, selector, target, seconds, repeat, delay) {

this.\_scheduler = scheduler;

this.\_target = target;

this.\_selector = selector;

this.setupTimerWithInterval(seconds, repeat, delay);

return true;

},

getSelector: function () {

return this.\_selector;

},

trigger: function () {

if (this.\_target && this.\_selector) {

this.\_target.call(this.\_selector, this.\_elapsed);

}

},

cancel: function () {

this.\_scheduler.unschedule(this.\_selector, this.\_target);

}

});

cc.TimerTargetCallback = cc.Timer.extend({

\_target: null,

\_callback: null,

\_key: null,

ctor: function () {

this.\_target = null;

this.\_callback = null;

},

initWithCallback: function (scheduler, callback, target, key, seconds, repeat, delay) {

this.\_scheduler = scheduler;

this.\_target = target;

this.\_callback = callback;

this.\_key = key;

this.setupTimerWithInterval(seconds, repeat, delay);

return true;

},

getCallback: function () {

return this.\_callback;

},

getKey: function () {

return this.\_key;

},

trigger: function () {

if (this.\_callback) this.\_callback.call(this.\_target, this.\_elapsed);

},

cancel: function () {

this.\_scheduler.unschedule(this.\_callback, this.\_target);

}

});

cc.Scheduler = cc.Class.extend({

\_timeScale: 1.0,

\_updatesNegList: null,

\_updates0List: null,

\_updatesPosList: null,

\_hashForTimers: null,

\_arrayForTimers: null,

\_hashForUpdates: null,

\_currentTarget: null,

\_currentTargetSalvaged: false,

\_updateHashLocked: false,

ctor: function () {

this.\_timeScale = 1.0;

this.\_updatesNegList = [];

this.\_updates0List = [];

this.\_updatesPosList = [];

this.\_hashForUpdates = {};

this.\_hashForTimers = {};

this.\_currentTarget = null;

this.\_currentTargetSalvaged = false;

this.\_updateHashLocked = false;

this.\_arrayForTimers = [];

},

\_schedulePerFrame: function (callback, target, priority, paused) {

var hashElement = this.\_hashForUpdates[target.\_\_instanceId];

if (hashElement && hashElement.entry) {

if (hashElement.entry.priority !== priority) {

if (this.\_updateHashLocked) {

cc.log("warning: you CANNOT change update priority in scheduled function");

hashElement.entry.markedForDeletion = false;

hashElement.entry.paused = paused;

return;

} else {

this.unscheduleUpdate(target);

}

} else {

hashElement.entry.markedForDeletion = false;

hashElement.entry.paused = paused;

return;

}

}

if (priority === 0) {

this.\_appendIn(this.\_updates0List, callback, target, paused);

} else if (priority < 0) {

this.\_priorityIn(this.\_updatesNegList, callback, target, priority, paused);

} else {

this.\_priorityIn(this.\_updatesPosList, callback, target, priority, paused);

}

},

\_removeHashElement: function (element) {

delete this.\_hashForTimers[element.target.\_\_instanceId];

cc.arrayRemoveObject(this.\_arrayForTimers, element);

element.Timer = null;

element.target = null;

element = null;

},

\_removeUpdateFromHash: function (entry) {

var self = this,

element = self.\_hashForUpdates[entry.target.\_\_instanceId];

if (element) {

cc.arrayRemoveObject(element.list, element.entry);

delete self.\_hashForUpdates[element.target.\_\_instanceId];

element.entry = null;

element.target = null;

}

},

\_priorityIn: function (ppList, callback, target, priority, paused) {

var self = this,

listElement = new cc.ListEntry(null, null, callback, target, priority, paused, false);

if (!ppList) {

ppList = [];

ppList.push(listElement);

} else {

var index2Insert = ppList.length - 1;

for (var i = 0; i <= index2Insert; i++) {

if (priority < ppList[i].priority) {

index2Insert = i;

break;

}

}

ppList.splice(i, 0, listElement);

}

self.\_hashForUpdates[target.\_\_instanceId] = new cc.HashUpdateEntry(ppList, listElement, target, null);

return ppList;

},

\_appendIn: function (ppList, callback, target, paused) {

var self = this,

listElement = new cc.ListEntry(null, null, callback, target, 0, paused, false);

ppList.push(listElement);

self.\_hashForUpdates[target.\_\_instanceId] = new cc.HashUpdateEntry(ppList, listElement, target, null, null);

},

setTimeScale: function (timeScale) {

this.\_timeScale = timeScale;

},

getTimeScale: function () {

return this.\_timeScale;

},

update: function (dt) {

this.\_updateHashLocked = true;

if (this.\_timeScale !== 1) dt \*= this.\_timeScale;

var i, list, len, entry;

for (i = 0, list = this.\_updatesNegList, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

for (i = 0, list = this.\_updates0List, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

for (i = 0, list = this.\_updatesPosList, len = list.length; i < len; i++) {

entry = list[i];

if (!entry.paused && !entry.markedForDeletion) entry.callback(dt);

}

var elt, arr = this.\_arrayForTimers;

for (i = 0; i < arr.length; i++) {

elt = arr[i];

this.\_currentTarget = elt;

this.\_currentTargetSalvaged = false;

if (!elt.paused) {

for (elt.timerIndex = 0; elt.timerIndex < elt.timers.length; ++(elt.timerIndex)) {

elt.currentTimer = elt.timers[elt.timerIndex];

elt.currentTimerSalvaged = false;

elt.currentTimer.update(dt);

elt.currentTimer = null;

}

}

if (this.\_currentTargetSalvaged && this.\_currentTarget.timers.length === 0) this.\_removeHashElement(this.\_currentTarget);

}

for (i = 0, list = this.\_updatesNegList; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

for (i = 0, list = this.\_updates0List; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

for (i = 0, list = this.\_updatesPosList; i < list.length;) {

entry = list[i];

if (entry.markedForDeletion) this.\_removeUpdateFromHash(entry);

else i++;

}

this.\_updateHashLocked = false;

this.\_currentTarget = null;

},

scheduleCallbackForTarget: function (target, callback\_fn, interval, repeat, delay, paused) {

this.schedule(callback\_fn, target, interval, repeat, delay, paused, target.\_\_instanceId + "");

},

schedule: function (callback, target, interval, repeat, delay, paused, key) {

var isSelector = false;

if (typeof callback !== "function") {

var selector = callback;

isSelector = true;

}

if (isSelector === false) {

if (arguments.length === 4 || arguments.length === 5) {

key = delay;

paused = repeat;

delay = 0;

repeat = cc.REPEAT\_FOREVER;

}

} else {

if (arguments.length === 4) {

paused = repeat;

repeat = cc.REPEAT\_FOREVER;

delay = 0;

}

}

if (key === undefined) {

key = target.\_\_instanceId + "";

}

cc.assert(target, cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget\_3);

var element = this.\_hashForTimers[target.\_\_instanceId];

if (!element) {

element = new cc.HashTimerEntry(null, target, 0, null, null, paused, null);

this.\_arrayForTimers.push(element);

this.\_hashForTimers[target.\_\_instanceId] = element;

} else {

cc.assert(element.paused === paused, "");

}

var timer, i;

if (element.timers == null) {

element.timers = [];

} else if (isSelector === false) {

for (i = 0; i < element.timers.length; i++) {

timer = element.timers[i];

if (callback === timer.\_callback) {

cc.log(cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget, timer.getInterval().toFixed(4), interval.toFixed(4));

timer.\_interval = interval;

return;

}

}

} else {

for (i = 0; i < element.timers.length; ++i) {

timer = element.timers[i];

if (timer && selector === timer.getSelector()) {

cc.log("CCScheduler#scheduleSelector. Selector already scheduled. Updating interval from: %.4f to %.4f", timer.getInterval(), interval);

timer.setInterval(interval);

return;

}

}

}

if (isSelector === false) {

timer = new cc.TimerTargetCallback();

timer.initWithCallback(this, callback, target, key, interval, repeat, delay);

element.timers.push(timer);

} else {

timer = new cc.TimerTargetSelector();

timer.initWithSelector(this, selector, target, interval, repeat, delay);

element.timers.push(timer);

}

},

scheduleUpdate: function (target, priority, paused) {

this.\_schedulePerFrame(function (dt) {

target.update(dt);

}, target, priority, paused);

},

\_getUnscheduleMark: function (key, timer) {

switch (typeof key) {

case "number":

case "string":

return key === timer.getKey();

case "function":

return key === timer.\_callback;

default:

return key === timer.getSelector();

}

},

unschedule: function (key, target) {

if (!target || !key) return;

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

var timers = element.timers;

for (var i = 0, li = timers.length; i < li; i++) {

var timer = timers[i];

if (this.\_getUnscheduleMark(key, timer)) {

if ((timer === element.currentTimer) && (!element.currentTimerSalvaged)) {

element.currentTimerSalvaged = true;

}

timers.splice(i, 1);

if (element.timerIndex >= i) {

element.timerIndex--;

}

if (timers.length === 0) {

if (self.\_currentTarget === element) {

self.\_currentTargetSalvaged = true;

} else {

self.\_removeHashElement(element);

}

}

return;

}

}

}

},

unscheduleUpdate: function (target) {

if (target == null) return;

var element = this.\_hashForUpdates[target.\_\_instanceId];

if (element) {

if (this.\_updateHashLocked) {

element.entry.markedForDeletion = true;

} else {

this.\_removeUpdateFromHash(element.entry);

}

}

},

unscheduleAllForTarget: function (target) {

if (target == null) {

return;

}

var element = this.\_hashForTimers[target.\_\_instanceId];

if (element) {

if (element.timers.indexOf(element.currentTimer) > -1 && (!element.currentTimerSalvaged)) {

element.currentTimerSalvaged = true;

}

element.timers.length = 0;

if (this.\_currentTarget === element) {

this.\_currentTargetSalvaged = true;

} else {

this.\_removeHashElement(element);

}

}

this.unscheduleUpdate(target);

},

unscheduleAll: function () {

this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

unscheduleAllWithMinPriority: function (minPriority) {

var i, element, arr = this.\_arrayForTimers;

for (i = arr.length - 1; i >= 0; i--) {

element = arr[i];

this.unscheduleAllForTarget(element.target);

}

var entry;

var temp\_length = 0;

if (minPriority < 0) {

for (i = 0; i < this.\_updatesNegList.length;) {

temp\_length = this.\_updatesNegList.length;

entry = this.\_updatesNegList[i];

if (entry && entry.priority >= minPriority) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updatesNegList.length) i++;

}

}

if (minPriority <= 0) {

for (i = 0; i < this.\_updates0List.length;) {

temp\_length = this.\_updates0List.length;

entry = this.\_updates0List[i];

if (entry) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updates0List.length) i++;

}

}

for (i = 0; i < this.\_updatesPosList.length;) {

temp\_length = this.\_updatesPosList.length;

entry = this.\_updatesPosList[i];

if (entry && entry.priority >= minPriority) this.unscheduleUpdate(entry.target);

if (temp\_length == this.\_updatesPosList.length) i++;

}

},

isScheduled: function (key, target) {

cc.assert(key, "Argument key must not be empty");

cc.assert(target, "Argument target must be non-nullptr");

var element = this.\_hashForUpdates[target.\_\_instanceId];

if (!element) {

return false;

}

if (element.timers == null) {

return false;

} else {

var timers = element.timers;

for (var i = 0; i < timers.length; ++i) {

var timer = timers[i];

if (key === timer.getKey()) {

return true;

}

}

return false;

}

},

pauseAllTargets: function () {

return this.pauseAllTargetsWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

pauseAllTargetsWithMinPriority: function (minPriority) {

var idsWithSelectors = [];

var self = this,

element, locArrayForTimers = self.\_arrayForTimers;

var i, li;

for (i = 0, li = locArrayForTimers.length; i < li; i++) {

element = locArrayForTimers[i];

if (element) {

element.paused = true;

idsWithSelectors.push(element.target);

}

}

var entry;

if (minPriority < 0) {

for (i = 0; i < this.\_updatesNegList.length; i++) {

entry = this.\_updatesNegList[i];

if (entry) {

if (entry.priority >= minPriority) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

}

if (minPriority <= 0) {

for (i = 0; i < this.\_updates0List.length; i++) {

entry = this.\_updates0List[i];

if (entry) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

for (i = 0; i < this.\_updatesPosList.length; i++) {

entry = this.\_updatesPosList[i];

if (entry) {

if (entry.priority >= minPriority) {

entry.paused = true;

idsWithSelectors.push(entry.target);

}

}

}

return idsWithSelectors;

},

resumeTargets: function (targetsToResume) {

if (!targetsToResume) return;

for (var i = 0; i < targetsToResume.length; i++) {

this.resumeTarget(targetsToResume[i]);

}

},

pauseTarget: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_pauseTarget);

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

element.paused = true;

}

var elementUpdate = self.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

elementUpdate.entry.paused = true;

}

},

resumeTarget: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_resumeTarget);

var self = this,

element = self.\_hashForTimers[target.\_\_instanceId];

if (element) {

element.paused = false;

}

var elementUpdate = self.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

elementUpdate.entry.paused = false;

}

},

isTargetPaused: function (target) {

cc.assert(target, cc.\_LogInfos.Scheduler\_isTargetPaused);

var element = this.\_hashForTimers[target.\_\_instanceId];

if (element) {

return element.paused;

}

var elementUpdate = this.\_hashForUpdates[target.\_\_instanceId];

if (elementUpdate) {

return elementUpdate.entry.paused;

}

return false;

},

scheduleUpdateForTarget: function (target, priority, paused) {

this.scheduleUpdate(target, priority, paused);

},

unscheduleCallbackForTarget: function (target, callback) {

this.unschedule(callback, target);

},

unscheduleUpdateForTarget: function (target) {

this.unscheduleUpdate(target);

},

unscheduleAllCallbacksForTarget: function (target) {

this.unschedule(target.\_\_instanceId + "", target);

},

unscheduleAllCallbacks: function () {

this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);

},

unscheduleAllCallbacksWithMinPriority: function (minPriority) {

this.unscheduleAllWithMinPriority(minPriority);

}

});

cc.Scheduler.PRIORITY\_SYSTEM = (-2147483647 - 1);

var initPlayerData = function () {

if (gamePlayerData == null) {

gamePlayerData = new Object();

gamePlayerData.soundSwitch = true;

gamePlayerData.money = 50;

gamePlayerData.currentLevel = 1;

gamePlayerData.levelStarData = [];

gamePlayerData.todayData = null;

gamePlayerData.gameRecordRemoveMonster = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

gamePlayerData.gameRemoveMonster = [0, 0, 0, 0, 0];

gamePlayerData.isCompleteDailyTsk = [false, false, false];

gamePlayerData.gameRankingListData = [];

gamePlayerData.fristEnterLevelOne = true;

gamePlayerData.bestScore = 0;

gamePlayerData.isJiHuo = 1;

}

var readData = readPlayerData();

for (var key in readData) {

gamePlayerData[key] = readData[key];

}

};

var Global = Global || {};

Global.gameType = 1;

Global.classicModeWidth = 7;

Global.classicModeHeight = 14;

Global.classicModeOffsetX = 0;

Global.classicModeOffsetY = 0;

Global.tileMapWidth = 90;

Global.tileMapHeight = 90;

Global.monsterWidth = 90;

Global.monsterHeight = 90;

Global.monsterFallSpeed = 70;

Global.createMonsterUpSpeed = 5;

Global.createMonsterUpTime = 8;

Global.createMonsterDownTime = 8;

Global.monsterShakeTimes = 50;

Global.gameOverData = [];

Global.currentMode = 1;

Global.currentSelectLevel = 1;

Global.maxSelectLevel = 100;

Global.monsterDown = false;

Global.jiHuoPrice = 0;

Global.jinBiShuArr = [0, 0, 0, 0];

Global.jinBiPriceArr = ["0", "0", "0", "0"];

Global.jieshuoPage = 3;

Global.isBaBa = 0;

Global.warningEffectID = 0;

Global.monsterSkin = [

[res.blue\_leftear\_png, res.blue\_rightear\_png, res.blue\_belly\_png, res.blue\_eyewhites\_png],

[res.green\_leftear\_png, res.green\_rightear\_png, res.green\_belly\_png, res.green\_eyewhites\_png],

[res.purple\_leftear\_png, res.purple\_rightear\_png, res.purple\_belly\_png, res.purple\_eyewhites\_png],

[res.red\_leftear\_png, res.red\_rightear\_png, res.red\_belly\_png, res.red\_eyewhites\_png],

[res.yellow\_leftear\_png, res.yellow\_rightear\_png, res.yellow\_belly\_png, res.yellow\_eyewhites\_png]

];

Global.monsterKingSkin = [

[res.big\_blue\_leftear\_png, res.big\_blue\_rightear\_png, res.big\_blue\_belly\_png, res.big\_blue\_number1\_png],

[res.big\_green\_leftear\_png, res.big\_green\_rightear\_png, res.big\_green\_belly\_png, res.big\_green\_number1\_png],

[res.big\_purple\_leftear\_png, res.big\_purple\_rightear\_png, res.big\_purple\_belly\_png, res.big\_purple\_number1\_png],

[res.big\_red\_leftear\_png, res.big\_red\_rightear\_png, res.big\_red\_belly\_png, res.big\_red\_number1\_png],

[res.big\_yellow\_leftear\_png, res.big\_yellow\_rightear\_png, res.big\_yellow\_belly\_png, res.big\_yellow\_number1\_png]

];

Global.music = [res.loginscene\_mp3, res.gamescene\_mp3, res.commonClickDown\_mp3, res.selectClickDown\_mp3, res.pause\_mp3, res.replay\_mp3, res.gamewin\_mp3, res.gamelose\_mp3, res.gameover\_mp3, res.toiletsucker\_mp3, res.dyeingbottle\_mp3, res.lightning\_mp3, res.flamebird\_mp3, res.ice\_mp3, res.box\_mp3, res.synthesis\_mp3, res.threemoreboom\_mp3, res.monsterBoom\_mp3, res.placemonster\_mp3, res.monstershake\_mp3, res.playmonsterthree\_mp3, res.playmonsterfour\_mp3, res.singlemonsterboomTwo\_mp3, res.starlight\_mp3, res.readygo\_mp3, res.warning\_mp3, res.completetask\_mp3, res.completeTarget\_mp3];

Global.monsterKingColor = ["blue", "green", "purple", "red", "yellow"];

Global.monsterSpecialEfects = [

["starlight", 0, 8],

["flame", 0, 7],

["lightning", 0, 6],

["blackHoleBoom", 0, 8]

];

Global.specialMonster = [res.ice\_png, res.timeMonster\_png, res.cage\_png, res.box\_png, res.advancedbox\_png, res.fwordbox\_png, res.fwordadvancedbox\_png];

Global.monsterDieEfects = [

["blackHole", 0, 6],

["blackSmoke", 1, 8],

["blue", 1, 7],

["green", 1, 7],

["purple", 1, 7],

["red", 1, 7],

["yellow", 1, 7],

["flameBoom", 0, 4],

["starlightBoom", 0, 5],

["timesBoom", 0, 5],

["lightningBoom", 0, 4],

["lightningBoom", 5, 9],

["iceBoom", 2, 5],

["woodenbox", 2, 7],

["ironbox", 2, 9],

["timemonsterboom", 0, 5],

["cageboom", 2, 9]

];

Global.gameTargetPicUrl = [res.GameTargetBlue\_png, res.GameTargetGreen\_png, res.GameTargetPurple\_png, res.GameTargetRed\_png, res.GameTargetYellow\_png, res.GameTargetIce\_png, res.GameTargetCage\_png, res.GameTargetBox\_png, res.GameTargetAdvancedBox\_png, res.GameTargetMonsterKing\_png, res.GameTargetScore\_png];

Global.dailyTaskMonster = [res.bluemonster\_png, res.greenmonster\_png, res.purplemonster\_png, res.redmonster\_png, res.yellowmonster\_png];

Global.gameRecordRemoveMonster = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

Global.fallCD = [10, 9, 8, 7, 6, 5, 4, 3];

Global.gameMaps = [

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 5, 3, 5, 0, 0],

[4, 5, 3, 2, 3, 5, 4],

[3, 1, 1, 4, 1, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 2, 0, 0, 0, 3, 3],

[2, 1, 1, 5, 1, 1, 2],

[4, 5, 3, 2, 3, 5, 4],

[9, 9, 1, 4, 1, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 0, 0, 0, 0, 0, 8],

[8, 8, 1, 5, 1, 8, 8],

[4, 5, 5, 2, 5, 5, 4],

[1, 2, 4, 3, 4, 2, 1]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 3, 6, 3, 0, 0],

[2, 1, 6, 1, 6, 1, 2],

[4, 5, 3, 6, 3, 5, 4],

[3, 3, 2, 5, 2, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 11, 0, 0, 0],

[0, 5, 5, 1, 2, 3, 0],

[4, 4, 3, 2, 3, 4, 4],

[3, 2, 1, 5, 1, 2, 3],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 0, 0, 2, 0, 0, 3],

[3, 0, 0, 2, 0, 0, 3],

[4, 0, 5, 1, 5, 0, 4],

[4, 5, 3, 2, 3, 5, 4],

[3, 1, 1, 4, 1, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[2, 0, 0, 0, 0, 0, 2],

[3, 3, 0, 0, 0, 3, 3],

[5, 4, 1, 0, 1, 4, 5],

[5, 4, 3, 5, 3, 4, 5],

[2, 2, 1, 1, 2, 2, 1]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[1, 2, 2, 5, 2, 2, 1],

[3, 6, 1, 3, 1, 6, 3],

[3, 4, 6, 5, 6, 4, 3],

[4, 1, 1, 4, 1, 1, 4]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 3, 0, 0, 0, 4, 4],

[1, 2, 1, 5, 1, 1, 3],

[4, 5, 6, 6, 6, 5, 4],

[1, 2, 3, 4, 5, 4, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[1, 2, 5, 3, 1, 2, 2],

[5, 1, 3, 1, 2, 4, 4],

[2, 2, 5, 11, 3, 5, 4],

[5, 3, 1, 5, 1, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 5, 3, 4, 3, 5, 3],

[2, 4, 1, 5, 1, 4, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[2, 1, 3, 1, 2, 4, 4],

[3, 4, 2, 2, 4, 1, 5],

[2, 4, 1, 5, 1, 3, 2],

[4, 5, 3, 2, 3, 5, 3],

[6, 2, 1, 6, 1, 2, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[4, 4, 1, 2, 3, 2, 1],

[6, 2, 3, 4, 3, 1, 6],

[6, 1, 2, 5, 5, 1, 6],

[6, 5, 3, 1, 3, 5, 6],

[3, 3, 2, 5, 2, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 0, 0, 0, 0, 0, 6],

[3, 3, 2, 5, 1, 4, 2],

[4, 8, 8, 8, 8, 8, 4],

[3, 2, 4, 5, 3, 3, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 3, 1, 2, 4, 2, 3],

[2, 1, 4, 5, 4, 3, 2],

[2, 1, 6, 6, 6, 4, 1]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 3, 6, 3, 6, 3, 0],

[0, 6, 6, 3, 6, 6, 0],

[1, 5, 6, 2, 6, 5, 4],

[5, 3, 3, 6, 3, 2, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 8, 0, 0, 0, 3, 5],

[8, 8, 0, 2, 0, 4, 2],

[2, 3, 1, 5, 2, 1, 2],

[1, 5, 2, 4, 3, 8, 8],

[2, 1, 1, 4, 1, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 2, 0, 9, 0, 4, 0],

[4, 2, 3, 8, 3, 2, 4],

[8, 8, 8, 8, 8, 8, 8],

[2, 3, 4, 9, 3, 5, 4],

[1, 2, 9, 9, 9, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 9, 9, 9, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[1, 0, 0, 0, 0, 0, 4],

[2, 1, 3, 8, 1, 2, 3],

[3, 8, 8, 8, 8, 8, 4],

[2, 3, 2, 8, 1, 5, 5]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 5, 0, 0, 0, 5, 0],

[0, 5, 0, 2, 0, 5, 0],

[4, 1, 1, 3, 3, 2, 4],

[6, 6, 4, 6, 5, 6, 6],

[11, 6, 1, 2, 3, 6, 11],

[6, 6, 1, 2, 3, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 3, 0, 0, 0, 3, 0],

[2, 7, 0, 0, 0, 7, 2],

[5, 3, 0, 3, 0, 3, 5],

[8, 4, 5, 2, 5, 4, 8],

[7, 8, 3, 2, 3, 8, 7],

[8, 7, 1, 8, 1, 7, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[1, 2, 5, 3, 5, 2, 1],

[1, 8, 8, 8, 8, 8, 1],

[2, 9, 9, 9, 9, 9, 2],

[4, 9, 7, 7, 7, 9, 4],

[3, 9, 9, 9, 9, 9, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 5, 0, 0, 0],

[1, 0, 4, 5, 4, 0, 1],

[6, 0, 3, 6, 3, 0, 6],

[2, 6, 3, 1, 3, 6, 2],

[4, 5, 6, 4, 6, 5, 4],

[3, 2, 2, 6, 2, 2, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 0, 0, 0, 0, 0, 9],

[0, 9, 0, 0, 0, 9, 0],

[0, 0, 9, 0, 9, 0, 0],

[5, 0, 0, 6, 0, 0, 3],

[3, 1, 6, 1, 6, 1, 1],

[4, 6, 2, 3, 2, 6, 4],

[6, 4, 4, 2, 4, 4, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 2, 0, 0, 0],

[0, 6, 0, 8, 0, 6, 0],

[0, 3, 6, 8, 6, 2, 0],

[1, 3, 2, 6, 4, 3, 1],

[2, 1, 6, 8, 6, 5, 2],

[2, 6, 7, 8, 7, 6, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 3, 1, 3, 5, 4, 3],

[8, 8, 8, 8, 8, 8, 8],

[5, 7, 8, 3, 8, 7, 5],

[7, 8, 3, 2, 3, 8, 7],

[8, 3, 2, 4, 1, 4, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 5, 4, 4, 2, 4, 3],

[6, 6, 6, 6, 6, 6, 6],

[3, 1, 1, 5, 1, 1, 3],

[6, 6, 6, 6, 6, 6, 6],

[2, 2, 1, 4, 1, 4, 4]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 9, 9, 9, 9, 9],

[9, 9, 0, 0, 0, 9, 9],

[0, 0, 0, 0, 0, 0, 0],

[6, 6, 6, 6, 3, 2, 4],

[4, 4, 5, 5, 4, 2, 1],

[3, 1, 2, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 10, 0, 10, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 1, 2, 1, 3, 4, 5],

[4, 5, 2, 5, 3, 4, 2],

[8, 8, 8, 8, 8, 8, 8],

[8, 8, 8, 8, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 10, 11, 10, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 8, 5, 5, 2, 6, 6],

[11, 8, 3, 3, 1, 6, 11],

[8, 8, 2, 2, 4, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 6, 6, 6, 0, 0],

[3, 0, 6, 3, 6, 0, 3],

[5, 2, 6, 2, 6, 4, 4],

[3, 1, 6, 6, 6, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 3, 0, 0, 0],

[0, 0, 5, 5, 1, 0, 0],

[0, 2, 1, 1, 3, 2, 0],

[8, 8, 6, 4, 6, 8, 8],

[8, 8, 6, 7, 6, 8, 8],

[8, 8, 6, 6, 6, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 0, 0, 0, 0, 0, 6],

[6, 6, 0, 0, 0, 6, 6],

[6, 6, 6, 0, 6, 6, 6],

[2, 1, 6, 1, 6, 1, 2],

[4, 5, 6, 2, 6, 5, 4],

[3, 3, 2, 5, 2, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 0, 9, 0, 9, 0, 9],

[0, 0, 0, 0, 0, 0, 0],

[8, 0, 0, 9, 0, 0, 8],

[8, 8, 0, 3, 0, 8, 8],

[8, 8, 8, 1, 8, 8, 8],

[4, 1, 1, 3, 4, 2, 4],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 11, 0, 0, 0],

[0, 0, 11, 8, 11, 0, 0],

[8, 8, 4, 3, 2, 1, 5],

[2, 1, 3, 1, 3, 2, 4],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 9, 0, 0, 0, 9, 0],

[0, 9, 0, 0, 0, 9, 0],

[0, 0, 9, 0, 9, 0, 0],

[0, 2, 0, 9, 0, 2, 0],

[1, 1, 5, 9, 2, 1, 4],

[4, 5, 3, 9, 1, 5, 4],

[3, 1, 1, 9, 3, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[9, 0, 9, 0, 9, 0, 9],

[0, 9, 0, 0, 0, 9, 0],

[0, 9, 0, 0, 0, 9, 0],

[9, 0, 9, 0, 9, 0, 9],

[0, 0, 0, 0, 0, 0, 0],

[2, 1, 6, 5, 6, 1, 2],

[6, 6, 4, 2, 3, 6, 6],

[2, 6, 2, 3, 4, 6, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 10, 0, 10, 0, 10, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 5, 0, 0, 0],

[2, 3, 5, 1, 4, 1, 2],

[4, 5, 3, 2, 3, 5, 4],

[3, 4, 2, 1, 2, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 10, 0, 10, 0, 0],

[0, 10, 0, 0, 0, 10, 0],

[8, 3, 2, 2, 1, 3, 8],

[8, 8, 10, 5, 10, 8, 8],

[4, 5, 8, 2, 8, 5, 4],

[4, 5, 8, 1, 8, 4, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 9, 0, 0, 0],

[0, 0, 0, 11, 0, 0, 0],

[6, 6, 8, 5, 8, 6, 6],

[11, 2, 4, 3, 1, 2, 11],

[3, 4, 6, 6, 6, 2, 1],

[1, 2, 4, 11, 5, 3, 5]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 0, 1, 0, 9, 9],

[9, 9, 4, 4, 2, 9, 9],

[9, 9, 6, 3, 6, 9, 9],

[6, 1, 2, 4, 3, 5, 6],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 10, 10, 10, 0, 0],

[0, 0, 0, 3, 0, 0, 0],

[4, 6, 6, 1, 6, 6, 4],

[1, 10, 10, 2, 10, 10, 2],

[3, 5, 2, 4, 5, 5, 3],

[3, 1, 10, 10, 10, 2, 5]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 3, 4, 2, 3, 4, 6],

[4, 6, 3, 6, 3, 6, 4],

[2, 1, 6, 1, 6, 1, 2],

[4, 3, 3, 6, 3, 2, 4],

[1, 2, 6, 5, 6, 5, 4]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 8, 0, 8, 0, 8, 0],

[8, 3, 8, 2, 8, 3, 8],

[2, 8, 1, 5, 1, 8, 2],

[4, 5, 5, 2, 5, 5, 4],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 10, 0, 0, 10, 0],

[0, 10, 0, 10, 0, 10, 0],

[9, 0, 0, 0, 0, 0, 9],

[0, 3, 11, 0, 11, 0, 9],

[3, 2, 6, 2, 6, 4, 2],

[1, 6, 5, 5, 4, 6, 3],

[9, 9, 9, 11, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[9, 0, 0, 0, 0, 0, 9],

[9, 9, 9, 9, 9, 9, 9],

[9, 0, 0, 0, 0, 0, 9],

[0, 0, 0, 0, 0, 0, 0],

[4, 4, 8, 8, 8, 2, 4],

[3, 1, 8, 7, 8, 4, 2],

[2, 5, 8, 5, 8, 5, 2],

[3, 1, 8, 7, 8, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 2, 0, 0, 0],

[0, 4, 3, 5, 2, 1, 0],

[0, 3, 1, 3, 4, 5, 0],

[6, 6, 6, 6, 6, 6, 6],

[6, 6, 8, 8, 8, 6, 6],

[6, 6, 8, 8, 8, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[2, 3, 2, 1, 1, 4, 5],

[3, 7, 3, 4, 3, 7, 7],

[6, 6, 6, 6, 6, 6, 6],

[6, 5, 3, 6, 3, 5, 6],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 0, 0, 0, 0, 0, 10],

[0, 0, 0, 0, 0, 0, 0],

[1, 0, 3, 2, 1, 0, 2],

[8, 0, 8, 4, 8, 0, 8],

[8, 8, 8, 5, 8, 8, 8],

[4, 5, 3, 2, 1, 5, 4],

[10, 10, 10, 10, 10, 10, 10]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 5, 0, 0, 0],

[0, 0, 2, 3, 1, 0, 0],

[0, 3, 4, 5, 4, 4, 0],

[0, 1, 3, 1, 3, 2, 0],

[0, 11, 11, 11, 11, 11, 0]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 6, 6, 6, 5, 3],

[0, 2, 6, 3, 2, 1, 5],

[0, 2, 6, 6, 6, 4, 4],

[4, 5, 3, 1, 6, 3, 2],

[3, 6, 6, 6, 6, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 2, 0, 2, 0, 0],

[0, 1, 6, 6, 6, 1, 0],

[2, 6, 6, 6, 6, 6, 2],

[4, 3, 10, 10, 10, 5, 4],

[2, 4, 10, 10, 10, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 0, 9, 0, 9, 0, 9],

[0, 0, 0, 6, 0, 0, 0],

[0, 0, 6, 7, 6, 0, 0],

[0, 0, 3, 6, 3, 0, 0],

[2, 1, 6, 1, 6, 1, 2],

[4, 5, 3, 6, 3, 5, 4],

[3, 1, 6, 9, 6, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 4, 2, 1, 3, 4, 0],

[8, 5, 5, 8, 5, 5, 8],

[8, 8, 8, 8, 8, 8, 8],

[6, 6, 6, 6, 6, 6, 6],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 0, 0, 4, 0, 0, 10],

[3, 10, 5, 10, 4, 10, 4],

[2, 1, 10, 0, 10, 2, 3],

[6, 6, 6, 6, 6, 6, 6],

[6, 11, 6, 11, 6, 11, 6],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 3, 0, 0, 0],

[0, 0, 2, 4, 1, 0, 0],

[6, 9, 9, 9, 1, 3, 6],

[2, 3, 5, 9, 5, 4, 2],

[4, 5, 3, 9, 3, 5, 4],

[6, 1, 1, 9, 9, 9, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 9, 9, 0, 9, 9, 0],

[0, 9, 9, 0, 9, 9, 0],

[5, 9, 9, 5, 9, 9, 5],

[4, 6, 6, 4, 6, 6, 4],

[4, 6, 6, 4, 6, 6, 4],

[3, 6, 6, 3, 6, 6, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 4, 5, 4, 0, 0],

[3, 2, 9, 6, 9, 2, 3],

[2, 1, 9, 6, 9, 1, 2],

[4, 5, 9, 6, 9, 5, 4],

[9, 9, 9, 6, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 5, 2, 2, 0, 0],

[8, 0, 3, 7, 3, 0, 8],

[8, 8, 1, 4, 1, 8, 8],

[8, 2, 5, 7, 3, 5, 8],

[8, 8, 2, 4, 1, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 10, 10, 10, 10],

[1, 0, 0, 0, 0, 0, 1],

[11, 3, 0, 0, 0, 2, 11],

[3, 4, 0, 0, 0, 1, 2],

[6, 1, 5, 2, 4, 2, 6],

[11, 6, 2, 3, 4, 6, 11]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 6, 6, 6, 0, 0],

[1, 4, 5, 5, 4, 3, 1],

[2, 3, 9, 9, 9, 2, 1],

[9, 9, 6, 6, 6, 9, 9],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 6, 0, 0, 0],

[0, 0, 6, 9, 6, 0, 0],

[0, 5, 9, 9, 9, 5, 0],

[6, 1, 1, 9, 2, 2, 6],

[4, 6, 3, 9, 3, 5, 4],

[2, 2, 4, 4, 3, 2, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 0, 0, 0, 0, 0, 0],

[4, 10, 0, 0, 0, 0, 4],

[5, 3, 10, 6, 3, 3, 2],

[2, 1, 6, 10, 6, 1, 2],

[6, 6, 6, 6, 10, 5, 3],

[2, 1, 4, 5, 2, 10, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8],

[4, 5, 8, 2, 8, 5, 4],

[1, 3, 5, 2, 3, 3, 4]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 0, 3, 6, 4, 0, 3],

[4, 0, 6, 11, 6, 0, 4],

[2, 6, 11, 11, 11, 6, 2],

[5, 6, 4, 2, 1, 6, 5],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 8, 0, 0, 0, 8, 0],

[3, 5, 1, 2, 3, 2, 1],

[3, 2, 5, 3, 5, 5, 4],

[4, 8, 8, 4, 8, 8, 4],

[1, 9, 9, 4, 9, 9, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0],

[0, 0, 2, 2, 3, 0, 0],

[6, 6, 6, 5, 8, 8, 8],

[6, 7, 3, 2, 4, 7, 8],

[6, 6, 6, 1, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 5, 1, 5, 0, 0],

[0, 5, 9, 9, 9, 5, 0],

[8, 8, 9, 9, 9, 8, 8],

[8, 8, 9, 9, 9, 8, 8],

[8, 8, 9, 9, 9, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 9, 9, 9, 0, 0],

[9, 9, 9, 9, 9, 9, 0],

[0, 0, 4, 0, 4, 0, 0],

[6, 0, 3, 2, 1, 0, 6],

[6, 6, 1, 3, 1, 6, 6],

[4, 5, 3, 2, 5, 5, 4],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 9, 0, 0, 0],

[0, 0, 9, 0, 9, 0, 0],

[0, 9, 0, 1, 0, 9, 0],

[0, 0, 3, 2, 3, 0, 0],

[8, 0, 4, 4, 3, 0, 8],

[8, 8, 1, 5, 1, 8, 8],

[6, 6, 6, 11, 6, 6, 6],

[6, 11, 6, 11, 6, 11, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 6, 6, 6, 6, 6, 6],

[9, 9, 9, 9, 9, 9, 9],

[6, 6, 0, 0, 0, 6, 6],

[1, 3, 1, 5, 1, 8, 8],

[4, 5, 5, 2, 5, 5, 4],

[3, 3, 1, 3, 1, 4, 5]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 10, 0, 0, 0, 0, 0],

[10, 6, 10, 2, 5, 3, 0],

[6, 6, 6, 10, 3, 5, 2],

[6, 6, 6, 6, 10, 4, 3],

[6, 6, 6, 6, 6, 10, 1],

[6, 6, 6, 6, 6, 6, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 10, 10, 0, 0, 0],

[10, 10, 0, 0, 0, 10, 10],

[0, 0, 0, 0, 0, 0, 0],

[1, 0, 0, 5, 0, 0, 1],

[9, 2, 0, 3, 0, 2, 9],

[9, 9, 1, 3, 1, 9, 9],

[9, 9, 5, 2, 5, 9, 9],

[9, 9, 5, 4, 3, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[4, 0, 0, 0, 0, 0, 4],

[8, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 4, 8, 8, 8],

[3, 5, 4, 2, 4, 5, 4],

[8, 2, 8, 8, 8, 1, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 0, 10, 10, 10],

[3, 0, 0, 0, 0, 0, 4],

[8, 3, 0, 0, 0, 4, 6],

[8, 8, 8, 1, 1, 6, 11],

[2, 5, 1, 2, 6, 11, 11],

[2, 3, 1, 6, 11, 11, 11]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 0, 0, 0, 0, 0, 0],

[6, 6, 2, 0, 0, 0, 0],

[6, 6, 6, 6, 2, 0, 0],

[6, 6, 6, 6, 6, 6, 3],

[5, 1, 1, 8, 3, 7, 2],

[1, 2, 5, 8, 7, 2, 3],

[8, 8, 8, 8, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 0, 10, 0, 10, 0, 10],

[0, 10, 0, 0, 0, 10, 0],

[1, 0, 0, 0, 0, 0, 1],

[6, 2, 6, 2, 6, 2, 3],

[6, 3, 6, 6, 6, 4, 5],

[6, 5, 6, 10, 6, 5, 4],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 10, 10, 10, 0, 0],

[0, 0, 0, 10, 0, 6, 5],

[8, 0, 0, 0, 0, 4, 8],

[8, 8, 1, 3, 1, 8, 8],

[8, 5, 3, 2, 5, 5, 8],

[3, 2, 10, 10, 10, 3, 1]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 6],

[0, 0, 0, 0, 2, 6, 8],

[2, 0, 0, 4, 6, 8, 8],

[1, 2, 4, 6, 2, 4, 3],

[4, 3, 6, 2, 1, 8, 8],

[4, 6, 5, 1, 3, 3, 1],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 10, 10, 10, 0],

[10, 11, 11, 11, 11, 10, 0],

[10, 10, 10, 10, 10, 10, 0],

[0, 0, 0, 0, 0, 0, 0],

[4, 0, 0, 0, 0, 0, 0],

[3, 2, 0, 0, 0, 0, 2],

[3, 1, 3, 0, 5, 3, 1]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 10, 10, 10, 10],

[10, 10, 10, 10, 10, 10, 10],

[0, 0, 0, 5, 0, 0, 0],

[0, 0, 1, 2, 1, 0, 0],

[0, 2, 6, 6, 6, 2, 0],

[1, 1, 2, 6, 2, 1, 1],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 0, 10, 10, 10],

[0, 0, 10, 0, 10, 0, 0],

[0, 0, 10, 0, 10, 0, 0],

[4, 0, 0, 4, 0, 0, 3],

[3, 2, 1, 5, 1, 1, 2],

[4, 3, 3, 5, 4, 5, 4],

[2, 2, 1, 2, 1, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 0, 0, 0, 9, 9],

[9, 9, 0, 0, 0, 9, 9],

[9, 9, 9, 0, 9, 9, 9],

[4, 0, 0, 0, 0, 0, 4],

[4, 0, 3, 6, 3, 0, 4],

[6, 1, 6, 1, 6, 1, 6],

[6, 5, 6, 6, 6, 5, 6],

[3, 3, 2, 6, 2, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 8, 8, 8, 8, 8, 8],

[9, 9, 9, 9, 9, 9, 9],

[0, 0, 0, 0, 0, 0, 0],

[2, 0, 5, 0, 5, 0, 1],

[4, 1, 1, 5, 1, 4, 2],

[9, 9, 9, 9, 9, 9, 9],

[8, 8, 8, 8, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[11, 11, 11, 11, 11, 0, 0],

[10, 10, 10, 10, 10, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[4, 0, 0, 0, 0, 0, 4],

[1, 3, 2, 5, 2, 2, 5],

[2, 1, 1, 2, 3, 3, 4],

[11, 3, 2, 4, 3, 2, 11]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 0, 10, 10, 10],

[6, 6, 6, 0, 6, 6, 6],

[10, 6, 10, 0, 10, 6, 10],

[0, 6, 0, 0, 0, 6, 0],

[0, 4, 0, 0, 0, 4, 0],

[4, 3, 4, 2, 2, 5, 4],

[4, 1, 1, 4, 1, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[8, 6, 8, 0, 8, 6, 8],

[6, 8, 6, 0, 6, 8, 6],

[8, 6, 8, 5, 8, 6, 8],

[2, 8, 1, 3, 1, 8, 3],

[8, 6, 8, 4, 8, 6, 8],

[2, 8, 1, 2, 1, 8, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 9, 9, 9, 9, 0],

[1, 3, 1, 3, 3, 1, 0],

[9, 9, 9, 9, 9, 9, 0],

[0, 0, 3, 4, 3, 0, 0],

[6, 6, 6, 1, 6, 6, 6],

[6, 5, 6, 6, 6, 5, 6],

[2, 1, 2, 4, 3, 2, 4]

],

[

[0, 0, 0, 0, 0, 0, 0],

[10, 0, 0, 0, 0, 0, 10],

[10, 0, 0, 0, 0, 0, 10],

[10, 0, 0, 0, 0, 0, 10],

[10, 0, 0, 0, 0, 0, 10],

[0, 0, 0, 0, 0, 0, 0],

[3, 4, 1, 0, 1, 3, 2],

[4, 5, 5, 0, 3, 2, 4],

[1, 2, 4, 1, 3, 1, 2]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[6, 6, 6, 6, 6, 6, 6],

[9, 9, 9, 9, 9, 9, 9],

[0, 0, 0, 0, 0, 0, 0],

[1, 2, 3, 0, 1, 3, 4],

[2, 3, 1, 0, 3, 2, 5],

[3, 2, 1, 0, 2, 4, 5],

[11, 11, 11, 0, 11, 11, 11]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 5, 2, 3, 10, 10],

[0, 0, 10, 10, 10, 0, 0],

[0, 0, 1, 4, 5, 0, 0],

[2, 5, 2, 2, 3, 5, 4],

[1, 2, 1, 4, 4, 1, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 6, 0, 0, 0, 6, 0],

[0, 10, 0, 0, 0, 10, 0],

[6, 0, 0, 6, 0, 0, 6],

[10, 0, 0, 10, 0, 0, 10],

[0, 0, 0, 0, 0, 0, 0],

[6, 2, 1, 6, 2, 1, 6],

[4, 10, 4, 2, 10, 5, 2],

[6, 2, 1, 6, 1, 3, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[9, 9, 9, 9, 9, 9, 9],

[8, 8, 8, 8, 8, 8, 8],

[9, 9, 9, 9, 9, 9, 9],

[0, 0, 0, 4, 0, 0, 0],

[2, 1, 4, 1, 6, 4, 2],

[8, 8, 8, 8, 8, 8, 8],

[9, 9, 9, 9, 9, 9, 9]

],

[

[0, 0, 0, 0, 0, 0, 0],

[6, 0, 0, 0, 0, 0, 6],

[8, 5, 0, 0, 0, 5, 8],

[6, 6, 0, 0, 0, 6, 6],

[8, 8, 5, 0, 5, 8, 8],

[6, 6, 6, 5, 6, 6, 6],

[8, 8, 8, 5, 8, 8, 8],

[6, 6, 6, 6, 6, 6, 6],

[8, 8, 8, 8, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[6, 6, 6, 6, 6, 6, 6],

[9, 9, 9, 9, 9, 9, 9],

[0, 0, 0, 0, 0, 0, 0],

[1, 0, 4, 0, 1, 0, 4],

[3, 0, 3, 0, 2, 0, 5],

[8, 11, 8, 0, 8, 11, 8],

[11, 11, 11, 11, 11, 11, 11],

[6, 6, 6, 6, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[10, 10, 10, 10, 10, 10, 10],

[10, 10, 10, 10, 10, 10, 10],

[0, 0, 0, 0, 0, 0, 0],

[4, 3, 3, 1, 3, 3, 1],

[6, 6, 6, 6, 6, 6, 6],

[6, 6, 6, 6, 6, 6, 6],

[8, 8, 8, 8, 8, 8, 8]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 6, 6, 6, 6, 6, 0],

[0, 6, 10, 10, 10, 0, 0],

[0, 6, 6, 6, 6, 0, 0],

[2, 6, 10, 10, 6, 1, 2],

[4, 6, 10, 10, 6, 5, 4],

[2, 6, 6, 6, 6, 3, 3]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 10, 0, 0, 0],

[0, 0, 10, 0, 10, 0, 0],

[0, 10, 0, 0, 0, 10, 0],

[10, 10, 0, 0, 0, 10, 10],

[2, 1, 1, 0, 0, 0, 0],

[6, 6, 6, 3, 6, 1, 2],

[4, 5, 3, 6, 3, 4, 3],

[3, 3, 2, 5, 2, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[1, 6, 1, 0, 3, 8, 1],

[6, 5, 6, 1, 8, 4, 8],

[5, 4, 3, 3, 1, 2, 5],

[6, 6, 6, 5, 6, 6, 6],

[4, 5, 5, 6, 5, 5, 4],

[6, 6, 6, 8, 6, 6, 6]

],

[

[0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0],

[3, 0, 0, 0, 0, 0, 3],

[6, 2, 0, 0, 0, 2, 6],

[11, 6, 2, 0, 2, 6, 11],

[6, 11, 6, 0, 6, 11, 6],

[8, 6, 8, 4, 8, 6, 8],

[8, 6, 8, 2, 8, 6, 8],

[6, 11, 6, 1, 6, 11, 6]

]

];

Global.gameTargetData = [

[150, 30, 50, 80, 5000, 0, 0, 0, 0, 10, 10],

[150, 30, 50, 80, 0, 2, 5, 8, 4, 5, 8],

[150, 30, 50, 80, 8000, 7, 6, 0, 0, 5, 8],

[150, 30, 50, 80, 0, 6, 4, 5, 8, 3, 5],

[100, 30, 50, 80, 0, 10, 1, 8, 7, 10, 8],

[100, 30, 50, 80, 0, 1, 10, 5, 12, 10, 10],

[100, 30, 50, 80, 8500, 0, 0, 2, 10, 10, 10],

[100, 30, 50, 80, 0, 4, 10, 6, 4, 10, 10],

[100, 30, 50, 80, 8500, 0, 0, 6, 3, 10, 10],

[120, 30, 50, 80, 12000, 0, 0, 10, 1, 10, 10],

[120, 30, 50, 80, 0, 4, 12, 3, 15, 5, 8],

[120, 30, 50, 80, 9500, 0, 0, 6, 3, 5, 8],

[120, 30, 50, 80, 0, 1, 15, 6, 6, 10, 8],

[150, 30, 50, 80, 0, 7, 5, 6, 2, 10, 10],

[140, 30, 50, 80, 0, 9, 1, 6, 3, 10, 10],

[130, 30, 50, 80, 9000, 0, 0, 6, 9, 8, 8],

[130, 30, 50, 80, 9500, 0, 0, 7, 8, 10, 8],

[120, 30, 50, 80, 0, 8, 5, 7, 8, 10, 8],

[120, 30, 50, 80, 0, 8, 3, 7, 7, 10, 8],

[120, 30, 50, 80, 0, 0, 0, 10, 2, 10, 10],

[140, 30, 50, 80, 10000, 0, 0, 7, 7, 8, 8],

[150, 30, 50, 80, 0, 8, 12, 7, 5, 10, 10],

[110, 30, 50, 80, 8000, 0, 0, 6, 8, 8, 8],

[130, 30, 50, 80, 0, 8, 6, 6, 7, 10, 10],

[140, 30, 50, 80, 0, 7, 4, 6, 9, 10, 10],

[140, 30, 50, 80, 0, 0, 0, 7, 13, 10, 10],

[140, 30, 50, 80, 12000, 0, 0, 6, 14, 5, 6],

[140, 30, 50, 80, 0, 8, 11, 6, 8, 5, 8],

[140, 30, 50, 80, 0, 9, 3, 7, 14, 8, 8],

[120, 30, 50, 80, 0, 0, 0, 10, 3, 10, 10],

[150, 30, 50, 80, 15000, 0, 0, 6, 10, 5, 8],

[150, 30, 50, 80, 0, 7, 12, 6, 7, 10, 8],

[200, 30, 50, 80, 18000, 0, 0, 6, 16, 10, 5],

[150, 30, 50, 80, 0, 7, 12, 8, 12, 8, 8],

[150, 30, 50, 80, 0, 0, 0, 10, 3, 8, 8],

[120, 30, 50, 80, 20000, 0, 0, 8, 10, 8, 8],

[130, 30, 50, 80, 0, 5, 12, 6, 8, 8, 8],

[150, 30, 50, 80, 22000, 0, 0, 9, 3, 8, 8],

[150, 30, 50, 80, 0, 9, 7, 7, 10, 8, 8],

[150, 30, 50, 80, 0, 0, 0, 10, 4, 8, 8],

[150, 30, 50, 80, 0, 8, 12, 6, 11, 8, 8],

[150, 30, 50, 80, 20000, 0, 0, 9, 10, 8, 8],

[150, 30, 50, 80, 18000, 0, 0, 6, 10, 5, 8],

[150, 30, 50, 80, 0, 7, 9, 8, 7, 3, 8],

[150, 30, 50, 80, 0, 9, 5, 10, 3, 8, 8],

[150, 30, 50, 80, 0, 8, 11, 7, 9, 8, 8],

[150, 30, 50, 80, 0, 7, 6, 6, 15, 8, 8],

[180, 30, 50, 80, 22000, 0, 0, 6, 17, 8, 8],

[150, 30, 50, 80, 0, 9, 9, 7, 10, 8, 8],

[150, 30, 50, 80, 0, 0, 0, 10, 5, 8, 5],

[120, 30, 45, 60, 24000, 0, 0, 6, 12, 8, 8],

[120, 30, 45, 60, 0, 6, 8, 9, 6, 8, 8],

[120, 30, 45, 60, 0, 6, 9, 8, 5, 8, 8],

[120, 30, 45, 60, 0, 7, 10, 6, 14, 8, 8],

[120, 30, 45, 60, 0, 9, 7, 10, 3, 8, 8],

[120, 30, 45, 60, 0, 6, 4, 8, 8, 8, 8],

[120, 30, 45, 60, 0, 8, 12, 6, 12, 8, 8],

[120, 30, 45, 60, 0, 8, 12, 6, 4, 8, 8],

[120, 30, 45, 60, 28000, 0, 0, 7, 12, 8, 8],

[120, 30, 45, 60, 30000, 0, 0, 10, 4, 8, 8],

[120, 30, 45, 60, 0, 6, 6, 8, 14, 8, 8],

[120, 30, 45, 60, 0, 6, 6, 8, 6, 8, 8],

[120, 30, 45, 60, 0, 9, 6, 6, 7, 8, 8],

[120, 30, 45, 60, 32000, 0, 0, 7, 11, 8, 8],

[120, 30, 45, 60, 0, 6, 7, 10, 4, 5, 8],

[120, 30, 45, 60, 40000, 0, 0, 0, 0, 8, 8],

[120, 30, 45, 60, 0, 6, 7, 7, 7, 8, 8],

[120, 30, 45, 60, 0, 8, 12, 7, 12, 8, 8],

[150, 30, 45, 60, 0, 8, 11, 6, 13, 8, 8],

[120, 30, 45, 60, 42000, 0, 0, 10, 4, 8, 8],

[120, 30, 45, 60, 0, 8, 7, 6, 11, 8, 8],

[150, 30, 50, 80, 0, 9, 6, 6, 19, 8, 8],

[150, 30, 50, 80, 0, 8, 14, 9, 6, 8, 8],

[150, 30, 50, 80, 38000, 0, 0, 7, 13, 8, 8],

[150, 30, 50, 80, 45000, 0, 0, 10, 6, 8, 8],

[150, 30, 50, 80, 0, 7, 9, 6, 13, 8, 8],

[150, 30, 50, 80, 0, 9, 7, 6, 17, 8, 8],

[150, 30, 50, 80, 0, 9, 8, 7, 8, 8, 8],

[150, 30, 50, 80, 0, 6, 13, 7, 5, 8, 8],

[120, 30, 45, 60, 0, 10, 4, 9, 14, 8, 8],

[120, 30, 45, 60, 0, 6, 11, 9, 14, 8, 8],

[120, 30, 45, 60, 42000, 0, 0, 9, 10, 8, 8],

[150, 30, 50, 80, 0, 8, 14, 6, 11, 8, 8],

[150, 30, 50, 80, 0, 7, 14, 8, 14, 8, 8],

[150, 30, 50, 80, 50000, 10, 7, 0, 0, 8, 8],

[150, 30, 50, 80, 0, 9, 10, 6, 10, 8, 8],

[150, 30, 50, 80, 0, 7, 18, 6, 10, 8, 8],

[120, 30, 45, 60, 0, 8, 12, 6, 11, 8, 8],

[120, 30, 45, 60, 50000, 9, 8, 0, 0, 8, 8],

[120, 30, 45, 60, 55000, 0, 0, 10, 6, 8, 8],

[120, 30, 45, 60, 52000, 9, 7, 0, 0, 5, 8],

[120, 30, 45, 60, 0, 6, 11, 9, 7, 5, 8],

[120, 30, 45, 60, 0, 8, 21, 7, 14, 5, 8],

[150, 30, 50, 80, 0, 6, 19, 7, 19, 5, 8],

[120, 30, 45, 60, 58000, 10, 9, 0, 0, 5, 8],

[150, 30, 50, 80, 0, 7, 7, 9, 14, 5, 8],

[120, 30, 45, 60, 0, 6, 18, 9, 7, 5, 8],

[150, 30, 50, 80, 0, 9, 9, 6, 7, 5, 8],

[120, 30, 45, 60, 0, 6, 16, 7, 4, 5, 8],

[120, 30, 45, 60, 60000, 10, 6, 0, 0, 5, 8]

];

var HelloWorldLayer = cc.Layer.extend({

sprite: null,

ctor: function () {

this.\_super();

var size = cc.winSize;

var helloLabel = new cc.LabelTTF("Hello World", "Arial", 38);

helloLabel.x = size.width / 2;

helloLabel.y = size.height / 2 + 200;

this.addChild(helloLabel, 5);

this.sprite = new cc.Sprite(res.HelloWorld\_png);

this.sprite.attr({

x: size.width / 2,

y: size.height / 2

});

this.addChild(this.sprite, 0);

return true;

}

});

var HelloWorldScene = cc.Scene.extend({

onEnter: function () {

this.\_super();

var layer = new HelloWorldLayer();

this.addChild(layer);

}

});

var res = {

Button\_Disable\_png: "res/Default/Button\_Disable.png",

youlose1: "res/loseLayer/youlose.json",

youlose2: "res/loseLayer/youlose.plist",

youlose3: "res/loseLayer/youlose.png",

youlose4: "res/loseLayer/youlose1.plist",

youlose5: "res/loseLayer/youlose1.png",

shengli\_json: "res/winLayer/shengli.json",

shengli\_png: "res/winLayer/shengli.png",

shengli\_plist: "res/winLayer/shengli.plist",

shengli1\_png: "res/winLayer/shengli1.png",

shengli1\_plist: "res/winLayer/shengli1.plist",

gameover\_json: "res/overLayer/gameover.json",

gameover\_plist: "res/overLayer/gameover.plist",

gameover\_png: "res/overLayer/gameover.png",

gameover1\_plist: "res/overLayer/gameover1.plist",

gameover1\_png: "res/overLayer/gameover1.png",

monster\_json: "res/monster/monster.json",

monster\_plist: "res/monster/monster.plist",

monster\_png: "res/monster/monster.png",

eye\_png: "res/monster/eye.png",

blue\_belly\_png: "res/monster/blue/belly.png",

blue\_eyewhites\_png: "res/monster/blue/eyewhites.png",

blue\_leftear\_png: "res/monster/blue/leftear.png",

blue\_rightear\_png: "res/monster/blue/rightear.png",

green\_belly\_png: "res/monster/green/belly.png",

green\_eyewhites\_png: "res/monster/green/eyewhites.png",

green\_leftear\_png: "res/monster/green/leftear.png",

green\_rightear\_png: "res/monster/green/rightear.png",

purple\_belly\_png: "res/monster/purple/belly.png",

purple\_eyewhites\_png: "res/monster/purple/eyewhites.png",

purple\_leftear\_png: "res/monster/purple/leftear.png",

purple\_rightear\_png: "res/monster/purple/rightear.png",

red\_belly\_png: "res/monster/red/belly.png",

red\_eyewhites\_png: "res/monster/red/eyewhites.png",

red\_leftear\_png: "res/monster/red/leftear.png",

red\_rightear\_png: "res/monster/red/rightear.png",

yellow\_belly\_png: "res/monster/yellow/belly.png",

yellow\_eyewhites\_png: "res/monster/yellow/eyewhites.png",

yellow\_leftear\_png: "res/monster/yellow/leftear.png",

yellow\_rightear\_png: "res/monster/yellow/rightear.png",

monsterKing\_json: "res/monsterKing/monsterKing.json",

monsterKing\_plist: "res/monsterKing/monsterKing.plist",

monsterKing\_png: "res/monsterKing/monsterKing.png",

big\_blue\_belly\_png: "res/monsterKing/blue/belly.png",

big\_blue\_leftear\_png: "res/monsterKing/blue/leftear.png",

big\_blue\_rightear\_png: "res/monsterKing/blue/rightear.png",

big\_blue\_number1\_png: "res/monsterKing/blue/number1.png",

big\_blue\_number2\_png: "res/monsterKing/blue/number2.png",

big\_blue\_number3\_png: "res/monsterKing/blue/number3.png",

big\_blue\_number4\_png: "res/monsterKing/blue/number4.png",

big\_blue\_number5\_png: "res/monsterKing/blue/number5.png",

big\_green\_belly\_png: "res/monsterKing/green/belly.png",

big\_green\_leftear\_png: "res/monsterKing/green/leftear.png",

big\_green\_rightear\_png: "res/monsterKing/green/rightear.png",

big\_green\_number1\_png: "res/monsterKing/green/number1.png",

big\_green\_number2\_png: "res/monsterKing/green/number2.png",

big\_green\_number3\_png: "res/monsterKing/green/number3.png",

big\_green\_number4\_png: "res/monsterKing/green/number4.png",

big\_green\_number5\_png: "res/monsterKing/green/number5.png",

big\_purple\_belly\_png: "res/monsterKing/purple/belly.png",

big\_purple\_leftear\_png: "res/monsterKing/purple/leftear.png",

big\_purple\_rightear\_png: "res/monsterKing/purple/rightear.png",

big\_purple\_number1\_png: "res/monsterKing/purple/number1.png",

big\_purple\_number2\_png: "res/monsterKing/purple/number2.png",

big\_purple\_number3\_png: "res/monsterKing/purple/number3.png",

big\_purple\_number4\_png: "res/monsterKing/purple/number4.png",

big\_purple\_number5\_png: "res/monsterKing/purple/number5.png",

big\_red\_belly\_png: "res/monsterKing/red/belly.png",

big\_red\_leftear\_png: "res/monsterKing/red/leftear.png",

big\_red\_rightear\_png: "res/monsterKing/red/rightear.png",

big\_red\_number1\_png: "res/monsterKing/red/number1.png",

big\_red\_number2\_png: "res/monsterKing/red/number2.png",

big\_red\_number3\_png: "res/monsterKing/red/number3.png",

big\_red\_number4\_png: "res/monsterKing/red/number4.png",

big\_red\_number5\_png: "res/monsterKing/red/number5.png",

big\_yellow\_belly\_png: "res/monsterKing/yellow/belly.png",

big\_yellow\_leftear\_png: "res/monsterKing/yellow/leftear.png",

big\_yellow\_rightear\_png: "res/monsterKing/yellow/rightear.png",

big\_yellow\_number1\_png: "res/monsterKing/yellow/number1.png",

big\_yellow\_number2\_png: "res/monsterKing/yellow/number2.png",

big\_yellow\_number3\_png: "res/monsterKing/yellow/number3.png",

big\_yellow\_number4\_png: "res/monsterKing/yellow/number4.png",

big\_yellow\_number5\_png: "res/monsterKing/yellow/number5.png",

SelectScene\_json: "res/SelectScene.json",

SelectItemNode\_json: "res/SelectItemNode.json",

Back\_png: "res/UI/SelectScene/Back.png",

Lock\_png: "res/UI/SelectScene/Lock.png",

LockBottom\_png: "res/UI/SelectScene/LockBottom.png",

NoPass\_png: "res/UI/SelectScene/NoPass.png",

PageViewDark\_png: "res/UI/SelectScene/PageViewDark.png",

PageViewLight\_png: "res/UI/SelectScene/PageViewLight.png",

Pass\_png: "res/UI/SelectScene/Pass.png",

SelectLvBg\_png: "res/UI/SelectScene/SelectLvBg.png",

Start\_png: "res/UI/SelectScene/Start.png",

StartDark\_png: "res/UI/SelectScene/StartDark.png",

StartLight\_png: "res/UI/SelectScene/StartLight.png",

StarOne\_png: "res/UI/SelectScene/StarOne.png",

StarTwo\_png: "res/UI/SelectScene/StarTwo.png",

StarThree\_png: "res/UI/SelectScene/StarThree.png",

dailyTaskLayer\_json: "res/dailyTaskLayer.json",

bottom\_png: "res/UI/DailyTask/bottom.png",

close1\_png: "res/UI/DailyTask/close1.png",

close2\_png: "res/UI/DailyTask/close2.png",

dailytask\_png: "res/UI/DailyTask/dailytask.png",

eliminate\_png: "res/UI/DailyTask/eliminate.png",

bottomDown\_png: "res/UI/DailyTask/bottomDown.png",

bottomUp\_png: "res/UI/DailyTask/bottomUp.png",

gold\_png: "res/UI/DailyTask/gold.png",

passLv\_png: "res/UI/DailyTask/passLv.png",

received\_png: "res/UI/DailyTask/received.png",

recesive\_png: "res/UI/DailyTask/recesive.png",

reward\_png: "res/UI/DailyTask/reward.png",

taskbotton\_png: "res/UI/DailyTask/taskbotton.png",

RankingListLayer\_json: "res/RankingListLayer.json",

RankingListNode\_json: "res/RankingListNode.json",

Bottom\_png: "res/UI/RankingList/bottom.png",

ImperialCrown\_png: "res/UI/RankingList/ImperialCrown.png",

LV\_png: "res/UI/RankingList/LV.png",

Medals\_png: "res/UI/RankingList/Medals.png",

RankingBottom\_png: "res/UI/RankingList/RankingBottom.png",

RankingList\_png: "res/UI/RankingList/RankingList.png",

score\_png: "res/UI/RankingList/score.png",

rankNumber\_plist: "res/UI/RankingList/rankNumber.plist",

rankNumber\_png: "res/UI/RankingList/rankNumber.png",

GamePauseLayer\_json: "res/GamePauseLayer.json",

continue\_png: "res/UI/GamePause/continue.png",

gamePauseBottom\_png: "res/UI/GamePause/gamePauseBottom.png",

help\_png: "res/UI/GamePause/help.png",

home\_png: "res/UI/GamePause/home.png",

music1\_png: "res/UI/GamePause/music1.png",

music2\_png: "res/UI/GamePause/music2.png",

pause\_png: "res/UI/GamePause/pause.png",

GameLoseLayer\_json: "res/GameLoseLayer.json",

resurrection\_png: "res/UI/GameLose/resurrection.png",

GameWinLayer\_json: "res/GameWinLayer.json",

GameOVerLayer\_json: "res/GameOVerLayer.json",

EndlessGameLayer\_json: "res/EndlessGameLayer.json",

LevelGameLayer\_json: "res/LevelGameLayer.json",

GameBarBotton\_png: "res/UI/GameScene/GameBarBotton.png",

GameBarCentral\_png: "res/UI/GameScene/GameBarCentral.png",

GameBarUp\_png: "res/UI/GameScene/GameBarUp.png",

GameBigGold\_png: "res/UI/GameScene/GameBigGold.png",

GameBotton\_png: "res/UI/GameScene/GameBotton.png",

GameChin\_png: "res/UI/GameScene/GameChin.png",

GameGoldAdd\_png: "res/UI/GameScene/GameGoldAdd.png",

GameGoldBotton\_png: "res/UI/GameScene/GameGoldBotton.png",

GameLV\_png: "res/UI/GameScene/GameLV.png",

GameScore\_png: "res/UI/GameScene/GameScore.png",

GameSmallGold\_png: "res/UI/GameScene/GameSmallGold.png",

GameTop2\_png: "res/UI/GameScene/GameTop2.png",

Pause\_png: "res/UI/GameScene/Pause.png",

pause2\_png: "res/UI/GameScene/pause2.png",

GameTarget\_png: "res/UI/GameScene/GameTarget.png",

GameTongue\_png: "res/UI/GameScene/GameTongue.png",

GameTooth\_png: "res/UI/GameScene/GameTooth.png",

GameTooth2\_png: "res/UI/GameScene/GameTooth2.png",

GameTop\_png: "res/UI/GameScene/GameTop.png",

GameTargetBotton\_png: "res/UI/GameScene/GameTargetBotton.png",

GameTargetBlue\_png: "res/UI/GameScene/GameTargetBlue.png",

GameTargetGreen\_png: "res/UI/GameScene/GameTargetGreen.png",

GameTargetPurple\_png: "res/UI/GameScene/GameTargetPurple.png",

GameTargetRed\_png: "res/UI/GameScene/GameTargetRed.png",

GameTargetYellow\_png: "res/UI/GameScene/GameTargetYellow.png",

GameTargetAdvancedBox\_png: "res/UI/GameScene/GameTargetAdvancedBox.png",

GameTargetBox\_png: "res/UI/GameScene/GameTargetBox.png",

GameTargetCage\_png: "res/UI/GameScene/GameTargetCage.png",

GameTargetIce\_png: "res/UI/GameScene/GameTargetIce.png",

GameTargetMonsterKing\_png: "res/UI/GameScene/GameTargetMonsterKing.png",

GameTargetScore\_png: "res/UI/GameScene/GameTargetScore.png",

gamescenebottom\_png: "res/UI/GameScene/gamescenebottom.png",

gamelevelupd\_png: "res/UI/GameScene/gamelevelupd.png",

gamelevelupdengji\_png: "res/UI/GameScene/gamelevelupdengji.png",

LoginScene\_json: "res/LoginScene.json",

ClassicMode\_png: "res/UI/LoginScene/ClassicMode.png",

ClassicMode1\_png: "res/UI/LoginScene/ClassicMode1.png",

LoginBottonUp\_png: "res/UI/LoginScene/LoginBottonUp.png",

eye1\_png: "res/UI/LoginScene/eye1.png",

eye2\_png: "res/UI/LoginScene/eye2.png",

LevelMode1\_png: "res/UI/LoginScene/LevelMode1.png",

LeveMode\_png: "res/UI/LoginScene/LeveMode.png",

LoginBotton\_png: "res/UI/LoginScene/LoginBotton.png",

LoginMusic\_png: "res/UI/LoginScene/LoginMusic.png",

LoginMusic1\_png: "res/UI/LoginScene/LoginMusic1.png",

LoginMusic2\_png: "res/UI/LoginScene/LoginMusic2.png",

LoginMusic3\_png: "res/UI/LoginScene/LoginMusic3.png",

LoginRank\_png: "res/UI/LoginScene/LoginRank.png",

LoginRank1\_png: "res/UI/LoginScene/LoginRank1.png",

LoginTask\_png: "res/UI/LoginScene/LoginTask.png",

LoginTask1\_png: "res/UI/LoginScene/LoginTask1.png",

LoginTop\_png: "res/UI/LoginScene/LoginTop.png",

MonsterBotton\_png: "res/UI/LoginScene/MonsterBotton.png",

monsterPuzzle\_png: "res/UI/LoginScene/monsterPuzzle.png",

Tongue\_png: "res/UI/LoginScene/Tongue.png",

ToothBotton\_png: "res/UI/LoginScene/ToothBotton.png",

blackHole\_plist: "res/dieAnimation/blackHole/blackHole.plist",

blackHole\_png: "res/dieAnimation/blackHole/blackHole.png",

blackHole1\_plist: "res/dieAnimation/blackHole/blackHole1.plist",

blackHole1\_png: "res/dieAnimation/blackHole/blackHole1.png",

blackHole2\_plist: "res/dieAnimation/blackHole/blackHole2.plist",

blackHole2\_png: "res/dieAnimation/blackHole/blackHole2.png",

blackHole3\_plist: "res/dieAnimation/blackHole/blackHole3.plist",

blackHole3\_png: "res/dieAnimation/blackHole/blackHole3.png",

blackHoleBoom\_plist: "res/dieAnimation/blackHoleBoom/blackHoleBoom.plist",

blackHoleBoom\_png: "res/dieAnimation/blackHoleBoom/blackHoleBoom.png",

blackSmoke\_plist: "res/dieAnimation/blackSmoke/blackSmoke.plist",

blackSmoke\_png: "res/dieAnimation/blackSmoke/blackSmoke.png",

blue\_plist: "res/dieAnimation/common/blue/blue.plist",

blue\_png: "res/dieAnimation/common/blue/blue.png",

green\_plist: "res/dieAnimation/common/green/green.plist",

green\_png: "res/dieAnimation/common/green/green.png",

purple\_plist: "res/dieAnimation/common/purple/purple.plist",

purple\_png: "res/dieAnimation/common/purple/purple.png",

red\_plist: "res/dieAnimation/common/red/red.plist",

red\_png: "res/dieAnimation/common/red/red.png",

yellow\_plist: "res/dieAnimation/common/yellow/yellow.plist",

yellow\_png: "res/dieAnimation/common/yellow/yellow.png",

flame\_plist: "res/dieAnimation/flame/flame.plist",

flame\_png: "res/dieAnimation/flame/flame.png",

flameBoom\_plist: "res/dieAnimation/flameBoom/flameBoom.plist",

flameBoom\_png: "res/dieAnimation/flameBoom/flameBoom.png",

lightning\_plist: "res/dieAnimation/lightning/lightning.plist",

lightning\_png: "res/dieAnimation/lightning/lightning.png",

starlight\_plist: "res/dieAnimation/starlight/starlight.plist",

starlight\_png: "res/dieAnimation/starlight/starlight.png",

starlightBoom\_plist: "res/dieAnimation/starlightBoom/starlightBoom.plist",

starlightBoom\_png: "res/dieAnimation/starlightBoom/starlightBoom.png",

starlightBoom2\_plist: "res/dieAnimation/starlightBoom/starlightBoom2.plist",

starlightBoom2\_png: "res/dieAnimation/starlightBoom/starlightBoom2.png",

timesBoom\_plist: "res/dieAnimation/timesBoom/timesBoom.plist",

timesBoom\_png: "res/dieAnimation/timesBoom/timesBoom.png",

dieCommonPlay\_plist: "res/dieAnimation/dieCommon/dieCommonPlay.plist",

dieCommonPlay\_png: "res/dieAnimation/dieCommon/dieCommonPlay.png",

lightningBoom\_plist: "res/dieAnimation/lightningBoom/lightningBoom.plist",

lightningBoom\_png: "res/dieAnimation/lightningBoom/lightningBoom.png",

iceBoom\_plist: "res/dieAnimation/iceBoom/iceBoom.plist",

iceBoom\_png: "res/dieAnimation/iceBoom/iceBoom.png",

woodenbox\_plist: "res/dieAnimation/woodenbox/woodenbox.plist",

woodenbox\_png: "res/dieAnimation/woodenbox/woodenbox.png",

ironbox\_plist: "res/dieAnimation/ironbox/ironbox.plist",

ironbox\_png: "res/dieAnimation/ironbox/ironbox.png",

timemonsterboom\_plist: "res/dieAnimation/timemonsterboom/timemonsterboom.plist",

timemonsterboom\_png: "res/dieAnimation/timemonsterboom/timemonsterboom.png",

cageboom\_plist: "res/dieAnimation/cageboom/cageboom.plist",

cageboom\_png: "res/dieAnimation/cageboom/cageboom.png",

dyeingbottle\_json: "res/props/dyeingbottle.json",

toiletplug\_json: "res/props/toiletplug.json",

props\_plist: "res/props/props.plist",

props\_png: "res/props/props.png",

bluemonster\_png: "res/UI/Icon/bluemonster.png",

greenmonster\_png: "res/UI/Icon/greenmonster.png",

purplemonster\_png: "res/UI/Icon/purplemonster.png",

redmonster\_png: "res/UI/Icon/redmonster.png",

yellowmonster\_png: "res/UI/Icon/yellowmonster.png",

dyebottle\_png: "res/UI/Icon/dyebottle.png",

dyebottlex\_png: "res/UI/Icon/dyebottlex.png",

toiletsucker\_png: "res/UI/Icon/toiletsucker.png",

toiletsuckerx\_png: "res/UI/Icon/toiletsuckerx.png",

advancedbox\_png: "res/specialMonster/advancedbox.png",

box\_png: "res/specialMonster/box.png",

cage\_png: "res/specialMonster/cage.png",

fwordadvancedbox\_png: "res/specialMonster/fwordadvancedbox.png",

fwordbox\_png: "res/specialMonster/fwordbox.png",

ice\_png: "res/specialMonster/ice.png",

timeMonster\_png: "res/specialMonster/timeMonster.png",

level\_id\_fnt: "res/level\_id.fnt",

level\_id\_0\_png: "res/level\_id\_0.png",

level\_star\_fnt: "res/level\_star.fnt",

level\_star\_0\_png: "res/level\_star\_0.png",

game\_Lv\_fnt: "res/game\_Lv.fnt",

game\_Lv\_0\_png: "res/game\_Lv\_0.png",

game\_score\_fnt: "res/game\_score.fnt",

game\_score\_0\_png: "res/game\_score\_0.png",

game\_gold\_fnt: "res/game\_gold.fnt",

game\_gold\_0\_png: "res/game\_gold\_0.png",

game\_goods\_fnt: "res/game\_goods.fnt",

game\_goods\_0\_png: "res/game\_goods\_0.png",

game\_time\_fnt: "res/game\_time.fnt",

game\_time\_0\_png: "res/game\_time\_0.png",

game\_win\_data\_fnt: "res/game\_win\_data.fnt",

game\_win\_data\_0\_png: "res/game\_win\_data\_0.png",

game\_lose\_data\_fnt: "res/game\_lose\_data.fnt",

game\_lose\_data\_0\_png: "res/game\_lose\_data\_0.png",

rank\_date\_fnt: "res/rank\_date.fnt",

rank\_date\_0\_png: "res/rank\_date\_0.png",

rank\_lv\_fnt: "res/rank\_lv.fnt",

rank\_lv\_0\_png: "res/rank\_lv\_0.png",

rank\_score\_fnt: "res/rank\_score.fnt",

rank\_score\_0\_png: "res/rank\_score\_0.png",

monsterscore\_fnt: "res/monsterscore.fnt",

monsterscore\_0\_png: "res/monsterscore\_0.png",

sdkgold\_fnt: "res/sdkgold.fnt",

sdkgold\_0\_png: "res/sdkgold\_0.png",

gamelevelup\_fnt: "res/gamelevelup.fnt",

gamelevelup\_0\_png: "res/gamelevelup\_0.png",

mic\_ttf: "res/mic.ttf",

comboPic\_plist: "res/comboPic.plist",

comboPic\_png: "res/comboPic.png",

encourage\_plist: "res/encourage.plist",

encourage\_png: "res/encourage.png",

loginscene\_mp3: "res/music/loginscene.mp3",

gamescene\_mp3: "res/music/gamescene.mp3",

commonClickDown\_mp3: "res/music/commonClickDown.mp3",

selectClickDown\_mp3: "res/music/selectClickDown.mp3",

pause\_mp3: "res/music/pause.mp3",

replay\_mp3: "res/music/replay.mp3",

gamewin\_mp3: "res/music/gamewin.mp3",

gamelose\_mp3: "res/music/gamelose.mp3",

gameover\_mp3: "res/music/gameover.mp3",

toiletsucker\_mp3: "res/music/toiletsucker.mp3",

dyeingbottle\_mp3: "res/music/dyeingbottle.mp3",

lightning\_mp3: "res/music/lightning.mp3",

flamebird\_mp3: "res/music/flamebird.mp3",

ice\_mp3: "res/music/ice.mp3",

box\_mp3: "res/music/box.mp3",

synthesis\_mp3: "res/music/synthesis.mp3",

threemoreboom\_mp3: "res/music/threemoreboom.mp3",

monsterBoom\_mp3: "res/music/monsterBoom.mp3",

placemonster\_mp3: "res/music/placemonster.mp3",

monstershake\_mp3: "res/music/monstershake.mp3",

playmonsterthree\_mp3: "res/music/playmonsterthree.mp3",

placemonsterfour\_mp3: "res/music/placemonsterfour.mp3",

singlemonsterboomTwo\_mp3: "res/music/singlemonsterboomTwo.mp3",

starlight\_mp3: "res/music/starlight.mp3",

readygo\_mp3: "res/music/readygo.mp3",

warning\_mp3: "res/music/warning.mp3",

completetask\_mp3: "res/music/completetask.mp3",

completeTarget\_mp3: "res/music/completeTarget.mp3",

beginnersguide\_png: "res/beginnersguide.png",

gameTargetLayer\_json: "res/gameTargetLayer.json",

gametargetbottom\_png: "res/UI/GameTarget/gametargetbottom.png",

gametargetimg\_png: "res/UI/GameTarget/gametargetimg.png",

readygo\_png: "res/UI/GameTarget/readygo.png",

TipsLayer\_json: "res/TipsLayer.json",

tips1\_png: "res/UI/Tips/tips1.png",

tips2\_png: "res/UI/Tips/tips2.png",

tips3\_png: "res/UI/Tips/tips3.png",

tips4\_png: "res/UI/Tips/tips4.png",

tips5\_png: "res/UI/Tips/tips5.png",

tips6\_png: "res/UI/Tips/tips6.png",

tips7\_png: "res/UI/Tips/tips7.png",

tipsbutton\_png: "res/UI/Tips/tipsbutton.png",

tooth1\_plist: "res/tooth1.plist",

tooth1\_png: "res/tooth1.png",

tooth2\_plist: "res/tooth2.plist",

tooth2\_png: "res/tooth2.png",

ActivationGameLayer\_json: "res/ActivationGameLayer.json",

SecondBuyLayer\_json: "res/SecondBuyLayer.json",

ShopLayer\_json: "res/ShopLayer.json",

PaymentResultLayer\_json: "res/PaymentResultLayer.json",

Paymentwaitlayer\_json: "res/Paymentwaitlayer.json",

sdk500pic\_png: "res/UI/Sdk/sdk500pic.png",

sdkanniu\_png: "res/UI/Sdk/sdkanniu.png",

sdkbou\_png: "res/UI/Sdk/sdkbou.png",

sdkcgv\_png: "res/UI/Sdk/sdkcgv.png",

sdkcgvbotttom2\_png: "res/UI/Sdk/sdkcgvbotttom2.png",

sdkcgvsmall\_png: "res/UI/Sdk/sdkcgvsmall.png",

sdkcgvsmall1\_png: "res/UI/Sdk/sdkcgvsmall1.png",

sdkcgvsmall2\_png: "res/UI/Sdk/sdkcgvsmall2.png",

sdkding\_png: "res/UI/Sdk/sdkding.png",

sdkding2\_png: "res/UI/Sdk/sdkding2.png",

sdkgou\_png: "res/UI/Sdk/sdkgou.png",

sdkjinb1\_png: "res/UI/Sdk/sdkjinb1.png",

sdkjinb2\_png: "res/UI/Sdk/sdkjinb2.png",

sdkjinb3\_png: "res/UI/Sdk/sdkjinb3.png",

sdkjinb4\_png: "res/UI/Sdk/sdkjinb4.png",

sdkx\_png: "res/UI/Sdk/sdkx.png",

timeUP\_png: "res/timeUP.png",

GameLevelUpNode\_json: "res/GameLevelUpNode.json"

};

var g\_resources = [];

for (var i in res) {

g\_resources.push(res[i]);

}

var GamePlayScene = GameScene.extend({

gameLayer: null,

gameLoseLayer: null,

gamePauseLayer: null,

gameControl: null,

upGameLayer: null,

gameRecordRemoveMonster: null,

gameWinLayer: null,

gameOverLayer: null,

gameTargetLayer: null,

tipsLayer: null,

sdkLayer: null,

onEnter: function () {

this.\_super();

this.init();

},

update: function (data) {

this.\_super(data);

this.gameControl.update(data);

},

init: function () {

this.gameRecordRemoveMonster = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

initSpriteFrame();

this.upGameLayer = new cc.Layer();

this.upGameLayer.x = 3;

this.upGameLayer.y = 180;

this.addChild(this.upGameLayer, 3);

this.gameControl = new GameControl();

this.gameControl.moveList = [];

this.gameControl.removeList = [];

this.gameControl.removeSpecialMonsterList = [];

this.gameControl.upMonsterList = [];

this.gameControl.downMonsterList = [];

this.initGameUI();

playBgMusic(1);

},

initGameUI: function () {

this.initGameLogicLayer();

this.initGameTargetLayer();

},

initGameTargetLayer: function () {

if (Global.currentMode == 1) {

this.gameTargetLayer = new GameTargetLayer();

this.addChild(this.gameTargetLayer, 4);

}

},

initGameLogicLayer: function () {

if (Global.currentMode == 1) {

Global.classicModeHeight = 14;

this.gameLayer = new LevelGameLayer();

}

if (Global.currentMode == 2) {

Global.classicModeHeight = 13;

this.gameLayer = new EndlessGameLayer();

}

this.addChild(this.gameLayer, 2);

this.gameControl.gameLayer = this.gameLayer;

if (Global.currentMode == 2) {

this.gameLayer.initMap();

}

},

gameLose: function () {

this.gameLoseLayer = new LoseLayer();

this.addChild(this.gameLoseLayer, 10);

},

gamePause: function () {

this.gamePauseLayer = new GamePauseLayer();

currentScene.addChild(this.gamePauseLayer, 10000);

},

gameWin: function (score, star) {

this.gameWinLayer = new WinLayer(score, star);

this.addChild(this.gameWinLayer, 10);

},

gameOver: function () {

this.gameOverLayer = new OverLayer();

this.addChild(this.gameOverLayer, 10);

},

gameTips: function (picID) {

this.tipsLayer = new TipsLayer(picID);

this.addChild(this.tipsLayer, 10);

},

createCombo: function (num) {

if (num > 0) {

var comboNum = new cc.Sprite("#" + "combo" + num + ".png");

comboNum.x = 360;

comboNum.y = 800 + 24;

this.addChild(comboNum, 8);

var comboPic = new cc.Sprite("#combo.png");

comboPic.x = 360 + 170;

comboPic.y = 800;

this.addChild(comboPic, 8);

var comboNumCallFun = cc.callFunc(function () {

comboNum.removeFromParent();

});

var comboPicCallFun = cc.callFunc(function () {

comboPic.removeFromParent();

});

comboNum.runAction(cc.sequence(cc.moveBy(0.4, cc.p(0, 50)), cc.fadeOut(0.2), comboNumCallFun));

comboPic.runAction(cc.sequence(cc.moveBy(0.4, cc.p(0, 50)), cc.fadeOut(0.2), comboPicCallFun));

}

},

createEncourage: function (score) {

var encouragePic = null;

if (score < 500) {

cc.log("score < 500");

} else if (score >= 500 && score < 1000) {

encouragePic = new cc.Sprite("#good.png");

} else if (score >= 1000 && score < 1500) {

encouragePic = new cc.Sprite("#great.png");

} else if (score >= 1500 && score < 2000) {

encouragePic = new cc.Sprite("#awesome.png");

} else {

encouragePic = new cc.Sprite("#verynice.png");

}

if (encouragePic != null) {

encouragePic.x = 360;

encouragePic.y = 900;

this.addChild(encouragePic, 8);

var encouragePicCallFun = cc.callFunc(function () {

encouragePic.removeFromParent();

});

encouragePic.runAction(cc.sequence(cc.moveBy(0.4, cc.p(0, 50)), cc.fadeOut(0.2), encouragePicCallFun));

}

},

createMonsterScore: function (monsterScore, pos) {

var monsterScoreTxt = new cc.LabelBMFont(monsterScore + "", res.monsterscore\_fnt);

monsterScoreTxt.setPosition(pos.x + 2, pos.y + 200);

this.addChild(monsterScoreTxt, 8);

var monsterScoreCallFun = cc.callFunc(function () {

monsterScoreTxt.removeFromParent();

});

monsterScoreTxt.runAction(cc.sequence(cc.moveBy(0.2, cc.p(0, 10)), cc.fadeOut(0.2), monsterScoreCallFun));

}

});

var LoginScene = GameScene.extend({

rootNode: null,

btnClassicMode: null,

btnLevelMode: null,

btnRank: null,

btnTask: null,

btnMusic: null,

btnMusicOff: null,

gameRankLayer: null,

gameTaskLayer: null,

saisircode: null,

sdkLayer: null,

onEnter: function () {

this.\_super();

engine.gameData.createJsonData();

this.init();

},

init: function () {

this.initUI();

this.initText();

this.initListener();

this.initSpriteFrame();

this.initTodayTask();

},

initText: function () {},

initTodayTask: function () {

var today = new Date();

if (gamePlayerData.todayData == null) {

gamePlayerData.todayData = new Object();

gamePlayerData.todayData.getMonth = today.getMonth();

gamePlayerData.todayData.getDate = today.getDate();

this.initTodayGameRemoveMonster();

} else {

if (today.getMonth() >= gamePlayerData.todayData.getMonth && today.getDate() > gamePlayerData.todayData.getDate) {

gamePlayerData.todayData.getMonth = today.getMonth();

gamePlayerData.todayData.getDate = today.getDate();

this.initTodayGameRemoveMonster();

}

}

savePlayerData();

},

initTodayGameRemoveMonster: function () {

gamePlayerData.gameRecordRemoveMonster = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

gamePlayerData.isCompleteDailyTsk = [false, false, false];

var monsterAType = randomNums(0, 5);

var monsterANum = randomNums(30, 51);

var monsterBType = randomNums(0, 5);

if (monsterAType == monsterBType) {

if (monsterBType != 1) {

monsterBType = monsterBType - 1;

} else {

monsterBType = monsterBType + 1;

}

}

var monsterBNum = randomNums(30, 51);

var monsterLevel = gamePlayerData.currentLevel + 3;

gamePlayerData.gameRemoveMonster[0] = monsterAType;

gamePlayerData.gameRemoveMonster[1] = monsterANum;

gamePlayerData.gameRemoveMonster[2] = monsterBType;

gamePlayerData.gameRemoveMonster[3] = monsterBNum;

gamePlayerData.gameRemoveMonster[4] = monsterLevel;

savePlayerData();

},

initUI: function () {

initSpriteFrame();

this.rootNode = loadCSS(res.LoginScene\_json);

this.addChild(this.rootNode);

this.btnClassicMode = this.rootNode.Button\_1;

this.btnClassicMode.tag = 1;

this.btnLevelMode = this.rootNode.Button\_2;

this.btnLevelMode.tag = 2;

this.btnRank = this.rootNode.Button\_3;

this.btnRank.tag = 3;

this.btnTask = this.rootNode.Button\_4;

this.btnTask.tag = 4;

this.btnMusic = this.rootNode.Button\_5;

this.btnMusic.setCascadeOpacityEnabled(true);

this.btnMusic.tag = 5;

this.btnMusicOff = this.rootNode.Button\_6;

this.btnMusicOff.setCascadeOpacityEnabled(true);

this.btnMusicOff.tag = 6;

this.saisircode = this.rootNode.Button\_7;

this.saisircode.setPressedActionEnabled(true);

this.saisircode.tag = 7;

this.saisircode.visible = false;

playBgMusic(0);

this.createEyeAction();

this.createTongueAction();

this.createMonster();

},

createEyeAction: function () {

var eyeBg = this.rootNode.Image\_3;

eyeBg.anchorX = 0.5;

eyeBg.anchorY = 0.4;

eyeBg.x = 200;

eyeBg.y = 1080;

eyeBg.runAction(cc.sequence(cc.scaleTo(0.08, 1, 0.7), cc.scaleTo(0.08, 1, 1), cc.delayTime(0.8)).repeatForever());

var eye = this.rootNode.Image\_4;

eye.anchorX = 0.5;

eye.anchorY = 0.5;

eye.x = 165;

eye.y = 1100;

var controlPoints1 = [cc.p(235, 1100), cc.p(200, 1070), cc.p(240, 1090)];

var controlPoints2 = [cc.p(165, 1100), cc.p(200, 1070), cc.p(160, 1090)];

var bezierAction1 = cc.bezierTo(0.16, controlPoints1);

var bezierAction2 = cc.bezierTo(0.16, controlPoints2);

eye.runAction(cc.sequence(bezierAction1, cc.delayTime(0.8), bezierAction2, cc.delayTime(0.8)).repeatForever());

},

createTongueAction: function () {

var tongue = this.rootNode.Image\_10;

tongue.anchorX = 0.5;

tongue.anchorY = 0;

tongue.x = 360;

tongue.y = 480;

tongue.runAction(cc.sequence(cc.rotateTo(0.3, 45), cc.delayTime(2), cc.rotateTo(0.3, -45), cc.delayTime(2)).repeatForever());

},

createMonster: function () {

for (var i = 1; i < 6; i++) {

var monster = new SpecialMonsterNode(i);

var sprite = this.rootNode.Panel\_1.getChildByName("Sprite\_" + i);

monster.x = sprite.x;

monster.y = sprite.y;

monster.monster.isStop = false;

this.rootNode.Panel\_1.addChild(monster);

}

},

initMusicTexture: function () {

if (gamePlayerData.soundSwitch) {

this.btnMusic.visible = true;

this.btnMusicOff.visible = false;

playBgMusic(0);

} else {

this.btnMusic.visible = false;

this.btnMusicOff.visible = true;

stopBgMusic(0);

}

savePlayerData();

},

initListener: function () {

this.btnClassicMode.addTouchEventListener(this.btnListenerFun, this.btnClassicMode);

this.btnLevelMode.addTouchEventListener(this.btnListenerFun, this.btnLevelMode);

this.btnRank.addTouchEventListener(this.btnListenerFun, this.btnRank);

this.btnTask.addTouchEventListener(this.btnListenerFun, this.btnTask);

this.btnMusic.addTouchEventListener(this.btnListenerFun, this.btnMusic);

this.btnMusicOff.addTouchEventListener(this.btnListenerFun, this.btnMusicOff);

this.saisircode.addTouchEventListener(this.btnListenerFun, this.saisircode);

},

btnListenerFun: function (sender, type) {

switch (sender.tag) {

case 1:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(3);

cc.log("btnClassicMode");

if (gamePlayerData.isJiHuo == 1) {

Global.currentMode = 2;

transitions(beginnersGuideScene);

} else {

createSDKLayerFun();

}

break;

}

break;

case 2:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(3);

cc.log("btnLevelMode");

Global.currentMode = 1;

transitions(SelectScene);

break;

}

break;

case 3:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnRank");

currentScene.creatGameRankLayer();

break;

}

break;

case 4:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnTask");

currentScene.creatGameTaskLayer();

break;

}

break;

case 5:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnMusic");

gamePlayerData.soundSwitch = false;

currentScene.initMusicTexture();

break;

}

break;

case 6:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

gamePlayerData.soundSwitch = true;

currentScene.initMusicTexture();

break;

}

break;

case 7:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

sendSMSFun();

break;

}

break;

}

savePlayerData();

},

creatGameRankLayer: function () {

this.gameRankLayer = new RankingListLayer();

currentScene.addChild(this.gameRankLayer, 1111, 1111);

},

creatGameTaskLayer: function () {

this.gameTaskLayer = new DailyTaskLayer();

currentScene.addChild(this.gameTaskLayer, 1111, 1111);

},

initSpriteFrame: function () {

cc.spriteFrameCache.addSpriteFrames(res.rankNumber\_plist);

}

});

var beginnersGuideScene = GameScene.extend({

beginnersGuideLayer: null,

onEnter: function () {

this.\_super();

this.init();

},

init: function () {

this.beginnersGuideLayer = new cc.Sprite(res.beginnersguide\_png);

this.beginnersGuideLayer.x = 360;

this.beginnersGuideLayer.y = 640;

this.addChild(this.beginnersGuideLayer);

var callFunAction = cc.callFunc(function () {

transitions(GamePlayScene);

});

this.runAction(cc.sequence(cc.delayTime(1.5), callFunAction));

}

});

var angleUnit = Math.PI / 180;

function randomNums(Min, Max) {

var Range = Max - Min;

var Rand = Math.random();

return (Min + Math.floor(Rand \* Range));

}

var shakeAction = function () {

var shakeDown = cc.moveBy(0.08, cc.p(0, -7));

var shakeUp = cc.moveBy(0.24, cc.p(0, 7));

return [shakeUp, shakeDown];

};

var monsterShakeAction = function (rowcol) {

var lx = currentScene.gameLayer.tileMapPos[rowcol[0]][rowcol[1]].x;

var ly = currentScene.gameLayer.tileMapPos[rowcol[0]][rowcol[1]].y;

var shakeDown = cc.moveTo(0.03, cc.p(lx - 1, ly - 5));

var shakeUp = cc.moveTo(0.04, cc.p(lx + 5, ly + 2));

var shakeLeft = cc.moveTo(0.02, cc.p(lx - 3, ly + 1));

var shakeRight = cc.moveTo(0.01, cc.p(lx + 2, ly - 4));

var shake1 = cc.moveTo(0.03, cc.p(lx + 0, ly + 2));

var shake2 = cc.moveTo(0.02, cc.p(lx + 4, ly - 3));

var shake3 = cc.moveTo(0.01, cc.p(lx - 3, ly - 2));

var shake4 = cc.moveTo(0.03, cc.p(lx + 2, ly - 5));

var shake5 = cc.moveTo(0.02, cc.p(lx, ly));

return [shakeUp, shakeDown, shakeLeft, shakeRight, shake1, shake2, shake3, shake4, shake5];

};

function createMonsterData(monster, j, i) {

var data = [];

data.push(monster);

data.push(j);

data.push(i);

return data;

}

function searchMaxColFromArr(arr) {

if (arr == null) return;

var length = arr.length;

var maxCol = 0;

var space = 0;

for (var i = 0; i < length; i++) {

if (arr[i] > space) {

space = arr[i];

maxCol = i;

}

}

return maxCol;

}

function getIDByName(name) {}

function playBgMusic(id) {

if (gamePlayerData.soundSwitch) {

cc.audioEngine.playMusic(Global.music[id], true);

}

}

function stopBgMusic(id) {

cc.audioEngine.stopMusic(Global.music[id]);

}

function playEffect(id) {

if (gamePlayerData.soundSwitch) {

cc.audioEngine.playEffect(Global.music[id]);

}

}

function playWarningEffect(id) {

if (gamePlayerData.soundSwitch) {

if (id == 25) {

Global.warningEffectID = cc.audioEngine.playEffect(Global.music[id], true);

} else {

cc.audioEngine.playEffect(Global.music[id], true);

}

}

}

function stopWarningEffect() {

cc.audioEngine.stopEffect(Global.warningEffectID);

}

function initSpriteFrame() {

cc.spriteFrameCache.addSpriteFrames(res.monster\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackHole\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackHole1\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackHole2\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackHole3\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackHoleBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blackSmoke\_plist);

cc.spriteFrameCache.addSpriteFrames(res.blue\_plist);

cc.spriteFrameCache.addSpriteFrames(res.green\_plist);

cc.spriteFrameCache.addSpriteFrames(res.purple\_plist);

cc.spriteFrameCache.addSpriteFrames(res.red\_plist);

cc.spriteFrameCache.addSpriteFrames(res.yellow\_plist);

cc.spriteFrameCache.addSpriteFrames(res.flame\_plist);

cc.spriteFrameCache.addSpriteFrames(res.flameBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.lightning\_plist);

cc.spriteFrameCache.addSpriteFrames(res.lightningBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.starlight\_plist);

cc.spriteFrameCache.addSpriteFrames(res.starlightBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.starlightBoom2\_plist);

cc.spriteFrameCache.addSpriteFrames(res.timesBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.dieCommonPlay\_plist);

cc.spriteFrameCache.addSpriteFrames(res.props\_plist);

cc.spriteFrameCache.addSpriteFrames(res.iceBoom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.woodenbox\_plist);

cc.spriteFrameCache.addSpriteFrames(res.ironbox\_plist);

cc.spriteFrameCache.addSpriteFrames(res.timemonsterboom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.cageboom\_plist);

cc.spriteFrameCache.addSpriteFrames(res.monsterKing\_plist);

cc.spriteFrameCache.addSpriteFrames(res.comboPic\_plist);

cc.spriteFrameCache.addSpriteFrames(res.encourage\_plist);

cc.spriteFrameCache.addSpriteFrames(res.rankNumber\_plist);

cc.spriteFrameCache.addSpriteFrames(res.tooth1\_plist);

cc.spriteFrameCache.addSpriteFrames(res.tooth2\_plist);

}

function addGameRankingListData(lv, date, score) {

var data = new Object();

data.lv = lv;

data.date = date;

data.score = score;

gamePlayerData.gameRankingListData.push(data);

}

function sortGameRankingListData() {

gamePlayerData.gameRankingListData.sort(function (a, b) {

return b.score - a.score

});

popGameRankingListData();

}

function popGameRankingListData() {

var length = gamePlayerData.gameRankingListData.length;

if (length <= 10) {

cc.log("gamePlayerData.gameRankingListData.length <= 10");

} else {

gamePlayerData.gameRankingListData.pop();

}

}

var TipsLayer = cc.Layer.extend({

rootNode: null,

ctor: function (picID) {

this.\_super();

this.init(picID);

},

init: function (picID) {

this.initUI(picID);

cc.director.pause();

},

initUI: function (picID) {

this.rootNode = loadCSS(res.TipsLayer\_json);

this.addChild(this.rootNode);

this.rootNode.Sprite\_1.initWithFile("res/UI/Tips/tips" + picID + ".png");

this.rootNode.Button\_1.setPressedActionEnabled(true);

this.rootNode.Button\_1.addTouchEventListener(this.btnDetermineFun);

if (picID == 3) {

this.rootNode.Button\_1.y = 100;

}

},

btnDetermineFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("Determine");

if (cc.director.isPaused()) {

cc.director.resume();

}

currentScene.tipsLayer.removeFromParent();

break;

}

}

});

var OverLayer = cc.Layer.extend({

rootNode: null,

home: null,

homePic: null,

replay: null,

replayPic: null,

gameover: null,

ctor: function () {

this.\_super();

this.init();

},

init: function () {

this.initUI();

this.initText();

this.initListener();

},

initUI: function () {

this.rootNode = loadCSS(res.GameOVerLayer\_json);

this.addChild(this.rootNode);

var animationPanel = this.rootNode.Panel\_3;

cc.spriteFrameCache.addSpriteFrames(res.gameover\_plist);

cc.spriteFrameCache.addSpriteFrames(res.gameover1\_plist);

multipleAnimation(res.gameover\_json, res.gameover\_json);

this.gameover = new GameSkeletalAnimation();

var data = skeletalAnimation[res.gameover\_json + "|gameover"];

data.playCount = 1;

this.gameover.initialize(data);

this.gameover.setFrameIntervalTime(24);

this.gameover.isRemoveFromComplete = false;

animationPanel.addChild(this.gameover);

this.gameover.x = 80;

this.gameover.y = 720;

this.home = new GameSkeletalAnimation();

var dataHome = skeletalAnimation[res.gameover\_json + "|gamehome"];

dataHome.playCount = 1;

this.home.initialize(dataHome);

this.home.setFrameIntervalTime(24);

this.home.isRemoveFromComplete = false;

animationPanel.addChild(this.home);

this.home.x = 260;

this.home.y = 420;

this.replay = new GameSkeletalAnimation();

var dataReplay = skeletalAnimation[res.gameover\_json + "|gamereplay"];

dataReplay.playCount = 1;

this.replay.initialize(dataReplay);

this.replay.setFrameIntervalTime(24);

this.replay.isRemoveFromComplete = false;

animationPanel.addChild(this.replay);

this.replay.x = 460;

this.replay.y = 420;

this.homePic = this.home.getBimapByUrl("#png19.png");

this.replayPic = this.replay.getBimapByUrl("#png20.png");

playEffect(8);

},

initText: function () {

this.rootNode.BitmapFontLabel\_1.setString(Global.gameOverData[0] + "");

this.rootNode.BitmapFontLabel\_2.setString(Global.gameOverData[1] + "");

this.rootNode.BitmapFontLabel\_3.setString(Global.gameOverData[2] + "");

},

initListener: function () {

engine.gameEvent.createMouseDown(this.homePic, this.btnHomeFun);

engine.gameEvent.createMouseDown(this.replayPic, this.btnReplayFun);

},

btnHomeFun: function () {

playEffect(2);

cc.log("btnHomeFun");

createCCSButtonScaling(this);

transitions(LoginScene);

},

btnReplayFun: function () {

playEffect(5);

cc.log("btnReplayFun");

createCCSButtonScaling(this);

transitions(GamePlayScene);

}

var LoseLayer = cc.Layer.extend({

rootNode: null,

resurrectionImg: null,

resurrectionImgPic: null,

goldText: null,

home: null,

homePic: null,

replay: null,

replayPic: null,

gamelose: null,

ctor: function () {

this.\_super();

this.init();

},

init: function () {

this.initUI();

var self = this;

var callFunAction = cc.callFunc(function () {

self.initText();

self.initListener();

});

this.runAction(cc.sequence(cc.delayTime(1), callFunAction));

},

initUI: function () {

this.rootNode = loadCSS(res.GameLoseLayer\_json);

this.addChild(this.rootNode);

this.rootNode.BitmapFontLabel\_1.visibie = false;

var animationPanel = this.rootNode.Panel\_2;

cc.spriteFrameCache.addSpriteFrames(res.youlose2);

cc.spriteFrameCache.addSpriteFrames(res.youlose4);

multipleAnimation(res.youlose1, res.youlose1);

this.gamelose = new GameSkeletalAnimation();

var data = skeletalAnimation[res.youlose1 + "|youlose"];

data.playCount = 1;

this.gamelose.initialize(data);

this.gamelose.setFrameIntervalTime(24);

this.gamelose.isRemoveFromComplete = false;

animationPanel.addChild(this.gamelose);

this.gamelose.x = 80;

this.gamelose.y = 720;

this.home = new GameSkeletalAnimation();

var dataHome = skeletalAnimation[res.youlose1 + "|gamehome"];

dataHome.playCount = 1;

this.home.initialize(dataHome);

this.home.setFrameIntervalTime(24);

this.home.isRemoveFromComplete = false;

animationPanel.addChild(this.home);

this.home.x = 260;

this.home.y = 420;

this.replay = new GameSkeletalAnimation();

var dataReplay = skeletalAnimation[res.youlose1 + "|gamereplay"];

dataReplay.playCount = 1;

this.replay.initialize(dataReplay);

this.replay.setFrameIntervalTime(24);

this.replay.isRemoveFromComplete = false;

animationPanel.addChild(this.replay);

this.replay.x = 460;

this.replay.y = 420;

this.resurrectionImg = new GameSkeletalAnimation();

var dataResurrectionImg = skeletalAnimation[res.youlose1 + "|gameresurrectionImg"];

dataResurrectionImg.playCount = 1;

this.resurrectionImg.initialize(dataResurrectionImg);

this.resurrectionImg.setFrameIntervalTime(24);

this.resurrectionImg.isRemoveFromComplete = false;

animationPanel.addChild(this.resurrectionImg);

this.resurrectionImg.x = 360;

this.resurrectionImg.y = 540;

this.homePic = this.home.getBimapByUrl("#youlosepng19.png");

this.replayPic = this.replay.getBimapByUrl("#youlosepng20.png");

this.resurrectionImgPic = this.resurrectionImg.getBimapByUrl("#youlosepng22.png");

playEffect(7);

},

initText: function () {

this.rootNode.BitmapFontLabel\_1.visibie = true;

},

initListener: function () {

engine.gameEvent.createMouseDown(this.homePic, this.btnHomeFun);

engine.gameEvent.createMouseDown(this.replayPic, this.btnReplayFun);

engine.gameEvent.createMouseDown(this.resurrectionImgPic, this.btnResurrectionFun);

},

showElement: function (obj) {},

btnHomeFun: function () {

playEffect(2);

cc.log("btnHomeFun");

createCCSButtonScaling(this);

transitions(SelectScene);

},

btnReplayFun: function () {

playEffect(5);

cc.log("btnReplayFun");

createCCSButtonScaling(this);

transitions(GamePlayScene);

},

btnResurrectionFun: function () {

playEffect(2);

cc.log("btnResurrectionFun");

createCCSButtonScaling(this);

if (gamePlayerData.money >= 300) {

gamePlayerData.money -= 300;

if (currentScene.gameLayer.isFullMonster) {

currentScene.gameLayer.isFullMonster = false;

var grid = [];

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < 7; j++) {

var monster = currentScene.gameLayer.tileMap.gridArr[i][j];

if (monster != null && monster.isMonster) {

grid = monster.rowcol;

break;

}

}

}

currentScene.gameLayer.useToiletPlug([grid[1], grid[0]]);

var callFunI = cc.callFunc(function () {

currentScene.gameLayer.startScheduleCheckout();

});

currentScene.gameLayer.runAction(cc.sequence(cc.delayTime(3), callFunI));

} else {

currentScene.gameLayer.startScheduleCheckout();

}

currentScene.gameLayer.countTime += 30;

currentScene.gameLayer.isGameOver = false;

currentScene.gameLoseLayer.removeFromParent();

currentScene.gameLoseLayer = null;

}

currentScene.gameLayer.refreshCurrentMoney();

savePlayerData();

}

});

});

var Mpos = new cc.p(0, 0);

var levelGameLayerTargetOffset = 120;

var LevelGameLayer = cc.Layer.extend({

rootNode: null,

btnPause: null,

btnAddGold: null,

btnToiletSucker: null,

btnDyeBottle: null,

panel: null,

touchPanel: null,

tileMap: null,

tileMapx: null,

tileMapPos: null,

tileMapUp: null,

tileMapDown: null,

tempMonsterArr: null,

tempMonsterArrs: null,

removeRows: null,

touchMonster: null,

isCreateMonsterDown: false,

isCreateCompleted: false,

tempColsArr: null,

isClearMonster: false,

isTouchDown: false,

touchPosition: null,

gameControl: null,

touchPosInBox: null,

frontGrid: null,

currentGrid: null,

isGenerateMonster: false,

connectMonsterArrs: null,

isTouchInBox: false,

isSelectDyeBottle: false,

isDyeing: false,

isSelectToiletSucker: false,

isPlayToilet: false,

isBanFall: false,

countTime: 0,

score: 0,

gameStarOne: 0,

gameStarTwo: 0,

gameStarThree: 0,

gameTargetScore: 0,

gameTargetOneID: 0,

gameTargetOneNum: 0,

gameTargetTwoID: 0,

gameTargetTwoNum: 0,

gameInitFallCD: 0,

gameFallCD: 0,

gameTarget: null,

isGameWin: false,

isGameOver: false,

isPause: false,

isTouchOnMouseUp: false,

isShowTipsTwo: false,

isShowTipsThree: false,

isShowGameAlarm: false,

fallToothTime: null,

isCanAddTimeMonster: false,

addTimeMonsterID: 0,

isFullMonster: false,

isTimeUP: false,

isPlayWarningEffect: false,

isPlayCompleteTargetOneEffect: false,

isPlayCompleteTargetTwoEffect: false,

isCanTouchDown: true,

isCanTouchDownCD: 0,

ctor: function () {

this.\_super();

this.init();

this.scheduleUpdate();

this.scheduleOnce(this.scheduleFallMonster, this.gameInitFallCD);

},

scheduleFun: function () {

this.refreshTime();

this.schedule(this.updateCheckout, 1);

},

update: function (dt) {

if (!this.isCanTouchDown) {

this.isCanTouchDownCD += dt;

if (this.isCanTouchDownCD > 0.06) {

this.isCanTouchDownCD = 0;

this.isCanTouchDown = true;

}

}

},

scheduleFallMonster: function () {

this.schedule(this.fallMonster, this.gameFallCD);

},

startScheduleCheckout: function () {

this.schedule(this.updateCheckout, 1);

},

stopScheduleCheckout: function () {

this.unschedule(this.updateCheckout);

},

init: function () {

if (Global.currentSelectLevel == 1) {

this.isShowTipsTwo = false;

this.isShowTipsThree = false;

}

this.initGameTargetData();

this.initUI();

this.initData();

this.initText();

this.initListener();

},

initGameTargetData: function () {

this.gameTarget = [];

this.countTime = Global.gameTargetData[Global.currentSelectLevel - 1][0];

this.gameStarOne = Global.gameTargetData[Global.currentSelectLevel - 1][1];

this.gameStarTwo = Global.gameTargetData[Global.currentSelectLevel - 1][2];

this.gameStarThree = Global.gameTargetData[Global.currentSelectLevel - 1][3];

this.gameTargetScore = Global.gameTargetData[Global.currentSelectLevel - 1][4];

this.gameTargetOneID = Global.gameTargetData[Global.currentSelectLevel - 1][5];

this.gameTargetOneNum = Global.gameTargetData[Global.currentSelectLevel - 1][6];

this.gameTargetTwoID = Global.gameTargetData[Global.currentSelectLevel - 1][7];

this.gameTargetTwoNum = Global.gameTargetData[Global.currentSelectLevel - 1][8];

this.gameInitFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][9];

this.gameFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][10];

this.calculateFallToothTime();

this.isCanAddTimeMonster = false;

this.addTimeMonsterID = 0;

},

initData: function () {

this.tempMonsterArrs = [];

this.removeRows = [];

this.connectMonsterArrs = [];

this.tempColsArr = [0, 0, 0, 0, 0, 0, 0];

this.tileMap = new TileMap();

var tileMapData = this.createTileMapData(Global.classicModeOffsetX, Global.classicModeOffsetY, Global.tileMapWidth, Global.tileMapHeight, Global.classicModeWidth, Global.classicModeHeight, null);

this.tileMap.initialize(tileMapData);

this.createMapPoint();

this.initMap(0);

},

createMapPoint: function () {

this.tileMapPos = [];

for (var row = 0; row < Global.classicModeHeight; row++) {

var arr = [];

for (var col = 0; col < Global.classicModeWidth; col++) {

var point = cc.p(col \* Global.tileMapWidth + Global.monsterWidth / 2 + 43, Global.tileMapHeight \* (row - 3) + Global.monsterHeight / 2);

arr.push(point);

}

this.tileMapPos.push(arr);

}

},

createTileMapData: function (offsetX, offsetY, gridWidth, gridHeight, gridMaxX, gridMaxY, defaultGridValue) {

var data = new Object();

data.offsetX = offsetX;

data.offsetY = offsetY;

data.gridWidth = gridWidth;

data.gridHeight = gridHeight;

data.gridMaxX = gridMaxX;

data.gridMaxY = gridMaxY;

data.defaultGridValue = defaultGridValue;

return data;

},

initUI: function () {

this.rootNode = loadCSS(res.LevelGameLayer\_json);

this.addChild(this.rootNode);

this.btnPause = this.rootNode.Button\_1;

this.btnAddGold = this.rootNode.Image\_8.Button\_2;

this.btnAddGold.setPressedActionEnabled(true);

this.btnDyeBottle = this.rootNode.Image\_9;

this.rootNode.dyebottlex\_1.visible = false;

this.btnToiletSucker = this.rootNode.Image\_12;

this.rootNode.toiletsuckerx\_2.visible = false;

this.rootNode.Image\_1.setLocalZOrder(-3);

this.rootNode.GameTongue\_2.setLocalZOrder(-2);

this.touchPanel = new cc.Layer();

this.touchPanel.width = 720;

this.touchPanel.height = 960;

this.touchPanel.x = 3;

this.touchPanel.y = 180;

this.rootNode.addChild(this.touchPanel, -1);

this.gameControl = currentScene.gameControl;

this.rootNode.GameAlarm\_1.visible = false;

this.initGameTargetUI();

this.refreshCurrentMoney();

this.refreshBtnAddMoney();

},

initGameTargetUI: function () {

if (this.gameTargetScore > 0) {

this.gameTarget.push(11);

this.gameTarget.push(this.gameTargetScore);

this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[10]);

this.rootNode.BitmapFontLabel\_5.setString(this.gameTargetScore + "");

if (this.gameTargetOneID > 0) {

this.gameTarget.push(this.gameTargetOneID);

this.gameTarget.push(this.gameTargetOneNum);

this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID - 1]);

this.rootNode.BitmapFontLabel\_5\_0.setString("0/" + this.gameTargetOneNum);

} else {

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.BitmapFontLabel\_5\_0.setString("0/" + this.gameTargetTwoNum);

} else {

this.rootNode.GameTargetGreen\_29.visible = false;

this.rootNode.BitmapFontLabel\_5\_0.visible = false;

this.rootNode.GameTargetBotton\_27.visible = false;

this.rootNode.GameTarget\_24.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBlue\_28.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBotton\_26.x += levelGameLayerTargetOffset;

this.rootNode.BitmapFontLabel\_5.x += levelGameLayerTargetOffset;

}

}

} else {

if (this.gameTargetOneID > 0) {

this.gameTarget.push(this.gameTargetOneID);

this.gameTarget.push(this.gameTargetOneNum);

this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID - 1]);

this.rootNode.BitmapFontLabel\_5.setString("0/" + this.gameTargetOneNum);

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.BitmapFontLabel\_5\_0.setString("0/" + this.gameTargetTwoNum);

} else {

this.rootNode.GameTargetGreen\_29.visible = false;

this.rootNode.BitmapFontLabel\_5\_0.visible = false;

this.rootNode.GameTargetBotton\_27.visible = false;

this.rootNode.GameTarget\_24.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBlue\_28.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBotton\_26.x += levelGameLayerTargetOffset;

this.rootNode.BitmapFontLabel\_5.x += levelGameLayerTargetOffset;

}

} else {

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.BitmapFontLabel\_5.setString("0/" + this.gameTargetTwoNum);

this.rootNode.GameTargetGreen\_29.visible = false;

this.rootNode.BitmapFontLabel\_5\_0.visible = false;

this.rootNode.GameTargetBotton\_27.visible = false;

this.rootNode.GameTarget\_24.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBlue\_28.x += levelGameLayerTargetOffset;

this.rootNode.GameTargetBotton\_26.x += levelGameLayerTargetOffset;

this.rootNode.BitmapFontLabel\_5.x += levelGameLayerTargetOffset;

} else {

this.rootNode.GameTargetGreen\_29.visible = false;

this.rootNode.BitmapFontLabel\_5\_0.visible = false;

this.rootNode.GameTargetBotton\_27.visible = false;

}

}

}

},

refreshBtnAddMoney: function () {

if (gamePlayerData.isJiHuo == 0) {

this.btnAddGold.visible = false;

}

if (gamePlayerData.isJiHuo == 1) {

this.btnAddGold.visible = true;

}

},

refreshTargetText: function () {

if (this.gameTarget[0] == 11) {

cc.log("第一个目标是分数");

if (this.gameTarget[2] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[2] - 1];

var str = tempnum + "/" + this.gameTarget[3];

this.rootNode.BitmapFontLabel\_5\_0.setString(str);

}

} else {

if (this.gameTarget[0] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[0] - 1];

var str = tempnum + "/" + this.gameTarget[1];

this.rootNode.BitmapFontLabel\_5.setString(str);

if (this.gameTarget[2] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[2] - 1];

var str = tempnum + "/" + this.gameTarget[3];

this.rootNode.BitmapFontLabel\_5\_0.setString(str);

}

}

}

},

refreshCurrentScore: function (recordConnectRemoveTimes, removeCount) {

var addScore = (recordConnectRemoveTimes \* 100 + 300) \* (removeCount - 2);

this.refreshCurrentScoreByScore(addScore);

currentScene.createEncourage(addScore);

},

refreshCurrentScoreByScore: function (score) {

if (!this.isGameWin && !this.isGameOver) {

this.score += score;

this.rootNode.Image\_4.BitmapFontLabel\_2.setString(this.score + "");

}

},

initText: function () {

this.rootNode.BitmapFontLabel\_1.setString("" + Global.currentSelectLevel);

this.rootNode.Image\_4.BitmapFontLabel\_2.setString("" + this.score);

this.rootNode.Image\_8.BitmapFontLabel\_3.setString("" + gamePlayerData.money);

},

refreshCurrentMoney: function () {

this.rootNode.Image\_8.BitmapFontLabel\_3.setString("" + gamePlayerData.money);

},

initListener: function () {

this.btnPause.addTouchEventListener(this.btnPauseFun, this.btnPause);

this.btnAddGold.addTouchEventListener(this.btnAddGoldFun, this.btnAddGold);

engine.gameEvent.createMouseDown(this.btnDyeBottle, this.btnDyeBottleFun);

engine.gameEvent.createMouseDown(this.btnToiletSucker, this.btnToiletSuckerFun);

this.addTouchListener();

},

btnDyeBottleFun: function () {

playEffect(2);

cc.log("btnDyeBottleFun");

if (currentScene.gameLayer.isSelectToiletSucker) return;

if (currentScene.gameLayer.isSelectDyeBottle) {

if (currentScene.gameLayer.isDyeing) {

cc.log("正在染色");

} else {

currentScene.gameLayer.isSelectDyeBottle = false;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = false;

}

} else {

if (gamePlayerData.money >= 500) {

currentScene.gameLayer.isSelectDyeBottle = true;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = true;

}

}

savePlayerData();

},

btnToiletSuckerFun: function () {

playEffect(2);

cc.log("btnToiletSuckerFun");

if (currentScene.gameLayer.isSelectDyeBottle) return;

if (currentScene.gameLayer.isSelectToiletSucker) {

if (currentScene.gameLayer.isPlayToilet) {

cc.log("正在播放马桶塞动画");

} else {

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

}

} else {

if (gamePlayerData.money >= 200) {

currentScene.gameLayer.isSelectToiletSucker = true;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = true;

}

}

savePlayerData();

},

btnPauseFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(4);

cc.log("btnPauseFun");

if (!this.isGameOver && !this.isGameWin) {

this.isPause = true;

cc.director.pause();

currentScene.gamePause();

}

break;

}

},

btnAddGoldFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnAddGoldFun");

createSDKLayerFun();

break;

}

},

initMap: function (row) {

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var gameMap = Global.gameMaps[Global.currentSelectLevel - 1];

var monsterType = gameMap[(9 - (i - 3)) - 1][j];

this.createMonster(i, j, monsterType);

}

}

},

createMonster: function (i, j, monsterType) {

if (monsterType <= 0) return;

var monster = new SpecialMonsterNode(monsterType, i, j);

if (monster != null) {

monster.x = this.tileMapPos[i][j].x;

monster.y = this.tileMapPos[i][j].y;

this.touchPanel.addChild(monster, Global.classicModeHeight - (i - 3));

monster.rowcol = [i, j];

monster.row = monster.rowcol[0];

this.tileMap.setGridInfo(i, j, monster);

}

},

fallMonstersCallFun: function () {

this.searchConnectedMonsters(this.tileMap);

var isConnected = this.checkIsConnectedMonsters();

if (isConnected) {

if (this.touchMonster == null) {

if (!this.isTouchDown) {

var isFallDowaning = this.searchIsFallDown(this.tileMap);

var isMonsterShake = this.searchIsShakeMonster(this.tileMap);

if (isFallDowaning && isMonsterShake) {

this.upInitMonster();

}

}

}

} else {

this.searchConnectedMonstersArrs();

this.shakeConnectedMonsters();

}

},

checkIsConnectedMonsters: function () {

if (this.tempMonsterArrs == null) return;

var connectedMonstersTimes = 0;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {

connectedMonstersTimes++;

}

}

}

}

if (connectedMonstersTimes == 0) {

return true;

} else {

return false;

}

},

initTileMapDown: function (row) {

this.createMonsterByRows(row, 2);

},

createMonsterByRows: function (rows, upAndDown) {

var monsterTypeArr = [];

for (var j = 0; j < rows; j++) {

for (var i = 0; i < 7; i++) {

var monsterType = randomNums(1, 8);

if (monsterType == 6) {

monsterType = randomNums(1, 6);

}

if (!this.isCanAddTimeMonster) {

if (monsterType == 7) {

monsterType = randomNums(1, 6);

}

} else {

if (monsterType == 7) {

this.isCanAddTimeMonster = false;

}

}

if (i >= 2) {

if (monsterTypeArr[i - 1] == monsterType && monsterTypeArr[i - 2] == monsterType) {

if (monsterType != 1) {

monsterType = monsterType - 1;

} else {

monsterType = 2;

}

}

}

monsterTypeArr.push(monsterType);

if (upAndDown == 1) {

if (this.tileMap.gridArr[Global.classicModeHeight - 1][i] == null && this.tileMap.gridArr[Global.classicModeHeight - 3][i] == null) {

var monster = new SpecialMonsterNode(monsterType, Global.classicModeHeight - 1, i);

monster.x = this.tileMapPos[Global.classicModeHeight - 1][i].x;

monster.y = this.tileMapPos[Global.classicModeHeight - 1][i].y;

this.touchPanel.addChild(monster);

this.tileMap.setGridInfo(Global.classicModeHeight - 1, i, monster);

}

}

if (upAndDown == 2) {

if (this.tileMap.gridArr[3 - 1 - j + 1][i] == null) {

var monster = new SpecialMonsterNode(monsterType, 3 - 1 - j + 1, i);

monster.x = this.tileMapPos[3 - 1 - j][i].x;

monster.y = this.tileMapPos[3 - 1 - j][i].y;

this.touchPanel.addChild(monster);

this.tileMap.setGridInfo(3 - 1 - j, i, monster);

this.gameControl.upMonster(3 - 1 - j, i, monster, rows);

}

}

}

}

},

monsterCallFun: function (target, data) {

if (data == null) return;

var monster = data[0];

var j = data[1];

var i = data[2];

monster.rowcol = [j, i];

monster.setLocalZOrder(Global.classicModeHeight - (j - 3));

monster.row = monster.rowcol[0];

this.tileMap.setGridInfo(j, i, monster);

},

searchRowsOfTileMap: function (tileMap, isFull) {

if (tileMap == null) return;

var tempNum = 0;

var num = 0;

if (isFull) {

for (var i = 3; i < tileMap.gridMaxY - 1; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

num++;

}

if (j == tileMap.gridMaxX - 1) {

if (num == tileMap.gridMaxX) {

tempNum++;

}

num = 0;

}

}

}

} else {

for (var \_i = 3; \_i < tileMap.gridMaxY - 1; \_i++) {

for (var \_j = 0; \_j < tileMap.gridMaxX; \_j++) {

if (tileMap.gridArr[\_i][\_j] != null) {

tempNum++;

break;

}

}

}

}

return tempNum;

},

searchColsOfTileMap: function (tileMap) {

if (tileMap == null) return;

var tempNum = [];

for (var i = 0; i < tileMap.gridMaxX; i++) {

if (tileMap.gridArr[tileMap.gridMaxY - 1 - 1][i] != null) {

tempNum.push(i);

}

}

return tempNum;

},

searchConnectedMonsters: function (tileMap) {

this.tempMonsterArrs = [];

var arr = [];

for (var i = 3; i < tileMap.gridMaxY - 2; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

this.tempMonsterArr = [];

if (tileMap.gridArr[i][j] != null) {

if (tileMap.gridArr[i][j].isMonster || tileMap.gridArr[i][j].isTimeMonster || tileMap.gridArr[i][j].isCageMonster || tileMap.gridArr[i][j].isBoxMonster || tileMap.gridArr[i][j].isAdvancedBoxMonster || tileMap.gridArr[i][j].isMonsterKing) {

if (!tileMap.gridArr[i][j].isSearched && !tileMap.gridArr[i][j].isWaiteDie && !tileMap.gridArr[i][j].isDown && !tileMap.gridArr[i][j].isUp) {

if (i == 3 || tileMap.gridArr[i - 1][j] != null || tileMap.gridArr[i][j].isBoxMonster || tileMap.gridArr[i][j].isAdvancedBoxMonster) {

tileMap.gridArr[i][j].isSearched = true;

this.tempMonsterArr.push(tileMap.gridArr[i][j]);

this.searchMonstersByDir(i, j, tileMap, 0, 1);

this.searchMonstersByDir(i, j, tileMap, 0, -1);

this.searchMonstersByDir(i, j, tileMap, -1, 0);

this.searchMonstersByDir(i, j, tileMap, 1, 0);

this.tempMonsterArrs.push(this.tempMonsterArr);

}

}

}

}

}

}

for (var i = 3; i < tileMap.gridMaxY - 2; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

tileMap.gridArr[i][j].isSearched = false;

}

}

}

},

searchMonstersByDir: function (i, j, tileMap, horizontal, vertical) {

if (i + vertical > 2 && i + vertical < 13) {

if (tileMap.isExistGrid(j + horizontal, i + vertical)) {

if (tileMap.gridArr[i + vertical][j + horizontal] != null) {

if (tileMap.gridArr[i + vertical][j + horizontal].isMonster || tileMap.gridArr[i + vertical][j + horizontal].isTimeMonster || tileMap.gridArr[i + vertical][j + horizontal].isCageMonster || tileMap.gridArr[i + vertical][j + horizontal].isBoxMonster || tileMap.gridArr[i + vertical][j + horizontal].isAdvancedBoxMonster || tileMap.gridArr[i + vertical][j + horizontal].isMonsterKing) {

if ((tileMap.gridArr[i + vertical][j + horizontal].skin == tileMap.gridArr[i][j].skin)) {

if (!tileMap.gridArr[i + vertical][j + horizontal].isSearched && !tileMap.gridArr[i + vertical][j + horizontal].isWaiteDie && !tileMap.gridArr[i + vertical][j + horizontal].isDown && !tileMap.gridArr[i + vertical][j + horizontal].isUp) {

if ((i + vertical == 3) || tileMap.gridArr[i + vertical - 1][j + horizontal] != null || tileMap.gridArr[i + vertical][j + horizontal].isBoxMonster || tileMap.gridArr[i + vertical][j + horizontal].isAdvancedBoxMonster) {

tileMap.gridArr[i + vertical][j + horizontal].isSearched = true;

this.tempMonsterArr.push(tileMap.gridArr[i + vertical][j + horizontal]);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 0, 1);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 0, -1);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, -1, 0);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 1, 0);

}

}

}

}

}

}

}

},

searchConnectedMonstersArrs: function () {

if (this.tempMonsterArrs == null) return;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {

this.connectMonsterArrs.push(this.tempMonsterArrs[i]);

}

}

}

}

},

shakeConnectedMonsters: function () {

if (this.connectMonsterArrs == null) return;

var connectMonsterArrsLength = this.connectMonsterArrs.length;

for (var i = 0; i < connectMonsterArrsLength; i++) {

var plength = this.connectMonsterArrs[i].length;

if (plength >= 3) {

var skin = this.connectMonsterArrs[i][0].skin;

var isExistArr = false;

var shakeTimes = Global.monsterShakeTimes;

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (this.connectMonsterArrs[i][j].isShake) {

var removeAct = this.gameControl.searchExistArrFromRemoveList(this.connectMonsterArrs[i][j]);

isExistArr = true;

break;

}

}

if (isExistArr) {

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (!this.connectMonsterArrs[i][j].isShake) {

if (removeAct != null) {

if (removeAct.arr != null) {

this.connectMonsterArrs[i][j].isShake = true;

playEffect(19);

this.connectMonsterArrs[i][j].playShakeAni();

removeAct.times = removeAct.times + 5;

removeAct.arr.push(this.connectMonsterArrs[i][j]);

playEffect(20);

}

}

}

}

} else {

var isMonsterRemove = false;

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (this.connectMonsterArrs[i][j].isRemve) {

this.connectMonsterArrs[i][j].isRemve = false;

isMonsterRemove = true;

}

}

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (!this.connectMonsterArrs[i][j].isShake && !this.connectMonsterArrs[i][j].isEnd && !this.connectMonsterArrs[i][j].isRemve) {

this.connectMonsterArrs[i][j].isShake = true;

this.connectMonsterArrs[i][j].playShakeAni();

if (j == plength - 1) {

playEffect(19);

}

}

}

this.gameControl.createRemoveList(this.connectMonsterArrs[i], shakeTimes, skin);

}

}

}

this.connectMonsterArrs = [];

},

removeConnectedMonsters: function () {

if (this.tempMonsterArrs == null) return;

this.tempColsArr = [0, 0, 0, 0, 0, 0, 0];

this.isClearMonster = true;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {}

}

}

}

this.tempMonsterArr = [];

this.tempMonsterArrs = [];

},

fallMonstersBySpace: function () {

Global.monsterDown = false;

if (this.isBanFall) return;

this.setFallRemoveRows();

for (var i = 0; i < this.removeRows.length; i++) {

var row = this.removeRows[i];

var space = 0;

for (var j = 3; j < Global.classicModeHeight; j++) {

var monster = this.tileMap.gridArr[j][row];

if (monster == null || monster.isBox || monster.isAdvancedBox) {

space++;

} else {

if (space > 0) {

if (!monster.isShake && !monster.isDown && !monster.isWaiteDie) {

if (monster.isMonster || monster.isIce || monster.isTimeMonster || monster.isCageMonster || monster.isMonsterKing) {

this.gameControl.moveMonster(j, row, monster, 1);

}

}

if (monster.isShake || monster.isBoxMonster || monster.isAdvancedBoxMonster || monster.isWaiteDie) {

space = 0;

}

}

}

}

}

this.removeRows = [];

if (!Global.monsterDown) {

if (this.gameControl.isMoveEnd) {

if (currentScene.gameLayer.isTouchOnMouseUp) {

currentScene.gameLayer.isTouchOnMouseUp = false;

}

var self = this;

var callAction = cc.callFunc(function () {

self.fallMonstersCallFun();

});

this.runAction(cc.sequence(cc.delayTime(0.2), callAction));

} else {

if (currentScene.gameLayer.isTouchOnMouseUp) {

currentScene.gameLayer.isTouchOnMouseUp = false;

var self = this;

var callAction = cc.callFunc(function () {

self.fallMonstersCallFun();

});

this.runAction(cc.sequence(cc.delayTime(0.2), callAction));

}

}

}

},

fallMonster: function () {

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver) return;

this.createMonsterByRows(1, 1);

this.fallMonstersBySpace();

},

upInitMonster: function () {

if (currentScene.gameLayer.isTouchDown || currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver) return;

if (currentScene.gameControl.moveList.length != 0 || currentScene.gameControl.removeList.length != 0 || currentScene.gameControl.removeSpecialMonsterList.length != 0 || currentScene.gameControl.upMonsterList.length > 1 || currentScene.gameControl.downMonsterList.length != 0) {

if (currentScene.gameControl.removeSpecialMonsterList.length != 0 || currentScene.gameControl.removeList.length != 0) {

return;

} else {

return;

}

}

if (!this.isCreateMonsterDown) {

this.printArr();

this.searchConnectedMonsters(this.tileMap);

var isConnected = this.checkIsConnectedMonsters();

if (!isConnected) {

this.searchConnectedMonstersArrs();

this.shakeConnectedMonsters();

} else {

var MonsterRowsCols = this.computeCreateMonsterDownRows(this.tileMap);

var rows = MonsterRowsCols[0];

if (rows == 0) return;

this.isCreateMonsterDown = true;

var cols = MonsterRowsCols[1];

var tileMapRows = this.searchRowsOfTileMap(this.tileMap, false);

for (var i = tileMapRows + 3; i >= 3; i--) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var boxCow = -1;

for (var m = 3; m < Global.classicModeHeight - 2; m++) {

if (this.tileMap.gridArr[m][j] != null) {

if (this.tileMap.gridArr[m][j].isAdvancedBox || this.tileMap.gridArr[m][j].isBox || this.tileMap.gridArr[m][j].isAdvancedBoxMonster || this.tileMap.gridArr[m][j].isBoxMonster) {

boxCow = m;

break;

}

}

}

if (boxCow == -1 || i < boxCow) {

if (i < Global.classicModeHeight - 2 && i + 1 <= Global.classicModeHeight - 3) {

if (this.tileMap.gridArr[i + 1][j] == null) {

var tileMapMonster = this.tileMap.gridArr[i][j];

if (tileMapMonster != null && !tileMapMonster.isBox && !tileMapMonster.isBoxMonster && !tileMapMonster.isAdvancedBox && !tileMapMonster.isAdvancedBoxMonster) {

if (!tileMapMonster.isDown && !tileMapMonster.isShake && !tileMapMonster.isWaiteDie && !tileMapMonster.isUp) {

this.gameControl.upMonster(i, j, tileMapMonster, rows);

}

}

}

}

}

}

}

this.createMonsterByRows(rows, 2);

}

}

},

getPosByGrid: function (Grid) {

return this.tileMapPos[Grid[1]][Grid[0]];

},

addTouchListener: function () {

engine.gameEvent.createMouseDown(this.touchPanel, this.onMouseDown);

engine.gameEvent.createMouseMove(this.touchPanel, this.onMouseMove);

engine.gameEvent.createMouseUp(this.touchPanel, this.onMouseUp);

},

onMouseDown: function (touch, event) {

currentScene.gameLayer.isTouchDown = true;

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver || currentScene.gameLayer.isBanFall) return;

var pos = touch.getLocation();

if (currentScene.gameLayer.touchMonster == null) {

currentScene.gameLayer.selectTouchMonster(pos);

}

return true;

},

selectTouchMonster: function (pos) {

if (!currentScene.gameLayer.isCanTouchDown) return;

currentScene.gameLayer.isCanTouchDown = false;

currentScene.gameLayer.touchPosition = pos;

if (!currentScene.gameLayer.checkTouchArea(pos)) return;

var tilePos = currentScene.gameLayer.getTileMapPosByPoint(pos);

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, tilePos);

grid[1] = grid[1] + 3;

if ((currentScene.gameLayer.isSelectDyeBottle || currentScene.gameLayer.isSelectToiletSucker) && !currentScene.gameLayer.isDyeing && !currentScene.gameLayer.isPlayToilet) {

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] == null && currentScene.gameLayer.isSelectToiletSucker) {

currentScene.gameLayer.isPlayToilet = false;

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

} else {

var props = new PropsNode();

if (currentScene.gameLayer.isSelectDyeBottle) {

props.type = 1;

currentScene.gameLayer.isDyeing = true;

props.initProps(res.dyeingbottle\_json, "dyeingbottle", currentScene.gameLayer, grid);

}

if (currentScene.gameLayer.isSelectToiletSucker) {

props.type = 2;

currentScene.gameLayer.isPlayToilet = true;

props.initProps(res.toiletplug\_json, "toiletplug", currentScene.gameLayer, grid);

}

var tempPos = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]];

props.setPosition(tempPos);

props.x = props.x + 3;

props.y = props.y + 180;

currentScene.gameLayer.addChild(props, 50);

}

} else {

currentScene.gameLayer.frontGrid = grid;

if (!currentScene.gameLayer.checkMonsterMove(grid)) {

currentScene.gameLayer.isTouchDown = true;

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isCreateBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

var monster = currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]], 1, tilePos);

currentScene.gameLayer.touchMonster = monster;

} else if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isCreateAdvanceBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

var monster = currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]], 2, tilePos);

currentScene.gameLayer.touchMonster = monster;

cc.log("grid[1] :" + grid[1] + " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " + "grid[0] :" + grid[0]);

} else {

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = true;

currentScene.gameLayer.touchMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];

}

}

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

return;

}

if (currentScene.gameLayer.touchMonster.isMonster || currentScene.gameLayer.touchMonster.isTimeMonster || currentScene.gameLayer.touchMonster.isMonsterKing) {

if (currentScene.gameLayer.touchMonster.isFalling) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

return;

}

if (currentScene.gameLayer.touchMonster.isShake) currentScene.gameLayer.touchMonster = null;

if (currentScene.gameLayer.touchMonster != null) {

currentScene.gameLayer.touchMonster.isDown = false;

currentScene.gameLayer.touchMonster.isMove = true;

if (!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] = null;

}

if (currentScene.gameLayer.removeRows.indexOf(grid[0]) == -1) {

currentScene.gameLayer.removeRows.push(grid[0]);

}

}

}

}

}

},

onMouseMove: function (touch, event) {

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver || currentScene.gameLayer.isBanFall) return;

var pos = touch.getLocation();

if (!currentScene.gameLayer.checkTouchArea(pos)) return;

if (currentScene.gameLayer.touchMonster == null) {

currentScene.gameLayer.selectTouchMonster(pos);

}

var delta = touch.getDelta();

var tilePos = currentScene.gameLayer.getTileMapPosByPoint(pos);

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) return;

var currentPoint = currentScene.gameLayer.touchMonster.getPosition();

var deltaX = tilePos.x - currentPoint.x;

var deltaY = tilePos.y - currentPoint.y;

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, tilePos);

grid[1] = grid[1] + 3;

currentScene.gameLayer.currentGrid = grid;

if (currentScene.gameLayer.touchPosition != null) {

if ((currentScene.gameLayer.currentGrid[0] != currentScene.gameLayer.frontGrid[0] || currentScene.gameLayer.currentGrid[1] != currentScene.gameLayer.frontGrid[1]) && (Math.abs(tilePos.x - currentScene.gameLayer.touchPosition.x) > Global.tileMapWidth)) {

currentScene.gameLayer.touchPosition = null;

currentScene.gameLayer.fallMonstersBySpace();

}

}

var monster = null;

if (grid[0] >= 0 && grid[0] < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 2) {

monster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];

}

var dirMonster = null;

if (currentScene.gameLayer.checkOutScreen(tilePos)) {

if (currentScene.gameLayer.checkMonsterMove(grid)) return;

if (monster != null) {

currentScene.gameLayer.isTouchInBox = true;

var tilePosDeltaX = tilePos.x - currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;

if (Math.abs(tilePosDeltaX) <= 15) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 2 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (grid[0] + 1 >= Global.classicModeWidth || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] - 1][grid[0]];

if (grid[1] - 1 > 2) {

if (dirMonster != null) {} else {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

}

if (tilePosDeltaX < 0 && Math.abs(tilePosDeltaX) > 15 && Math.abs(tilePosDeltaX) < 45) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 2 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (grid[0] + 1 >= Global.classicModeWidth || dirMonster != null) {

cc.log("上左右 都已满");

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] - 1][grid[0]];

if (grid[1] - 1 > 2) {

if (dirMonster != null) {

cc.log("上左右下 都已满");

} else {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

}

if (tilePosDeltaX > 0 && Math.abs(tilePosDeltaX) > 15 && Math.abs(tilePosDeltaX) < 45) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (grid[0] + 1 >= Global.classicModeWidth || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 2 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

cc.log("上左右 都已满");

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] - 1][grid[0]];

if (grid[1] - 1 > 2) {

if (dirMonster != null) {

cc.log("上左右下 都已满");

} else {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] - 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

}

currentScene.gameLayer.touchMonster.setPosition(tilePos);

} else {

if (currentScene.gameLayer.isTouchInBox) {

currentScene.gameLayer.isTouchInBox = false;

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].y;

}

} else {

if (deltaX > 0) {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 1, 0)) {

tilePos.x = currentPoint.x;

}

} else {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, -1, 0)) {

tilePos.x = currentPoint.x;

}

}

if (deltaY > 0) {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 0, 1)) {

tilePos.y = currentPoint.y;

}

} else {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 0, -1)) {

tilePos.y = currentPoint.y;

}

}

}

currentScene.gameLayer.touchMonster.setPosition(tilePos);

}

}

},

onMouseUp: function (touch, event) {

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) {

currentScene.gameLayer.touchMonster = null;

currentScene.gameLayer.currentGrid = null;

currentScene.gameLayer.frontGrid = null;

return;

}

var currentPoint = currentScene.gameLayer.touchMonster.getPosition();

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, currentPoint);

grid[1] = grid[1] + 3;

var gridPos = currentScene.gameLayer.getPosByGrid(grid);

if (currentScene.gameLayer.touchMonster != null) {

currentScene.gameLayer.isTouchDown = false;

currentScene.gameLayer.touchMonster.isMove = false;

currentScene.gameLayer.touchMonster.setPosition(gridPos);

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox) {

var skin = currentScene.gameLayer.touchMonster.skin;

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin + 1);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin = skin;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster = true;

} else if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox) {

var skin = currentScene.gameLayer.touchMonster.skin;

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin + 1);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin = skin;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster = true;

cc.log("grid[1] :" + grid[1] + " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " + "grid[0] :" + grid[0]);

} else {

if (grid[0] - 1 >= 0 && grid[0] - 1 < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 2 && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].rowcol = [grid[1], grid[0] - 1];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

if (grid[0] + 1 >= 0 && grid[0] + 1 < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 2 && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].rowcol = [grid[1], grid[0] + 1];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

if (grid[0] >= 0 && grid[0] < Global.classicModeWidth && grid[1] + 1 >= 3 && grid[1] + 1 < Global.classicModeHeight - 2 && currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]]);

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].rowcol = [grid[1] + 1, grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].setLocalZOrder(Global.classicModeHeight - (grid[1] + 1 - 3));

} else {

var isHaveEmptyGrid = false;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

if (currentScene.gameLayer.tileMap.gridArr[i][j] == null && !isHaveEmptyGrid) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[i][j]);

currentScene.gameLayer.tileMap.gridArr[i][j] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[i][j].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[i][j].rowcol = [i, j];

currentScene.gameLayer.tileMap.gridArr[i][j].row = currentScene.gameLayer.tileMap.gridArr[i][j].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[i][j].setLocalZOrder(Global.classicModeHeight - (i + 1 - 3));

isHaveEmptyGrid = true;

}

}

}

if (!isHaveEmptyGrid) {

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.touchMonster = null;

}

}

}

}

}

}

}

currentScene.gameLayer.isTouchOnMouseUp = true;

currentScene.gameLayer.fallMonstersBySpace();

currentScene.gameLayer.touchMonster = null;

currentScene.gameLayer.currentGrid = null;

currentScene.gameLayer.frontGrid = null;

currentScene.gameLayer.isTouchDown = false;

},

setFallRemoveRows: function () {

this.removeRows = [];

for (var i = 0; i < Global.classicModeWidth; i++) {

this.removeRows.push(i);

}

},

checkTouchArea: function (pos) {

if (pos.x < 718 && pos.x > 2 && pos.y < 960 && pos.y > 180) {

return true;

} else {

return false;

}

},

getTileMapGirdByPoint: function (tileMap, pos) {

var posX = pos.x - 43;

var posY = pos.y;

var lx = parseInt(posX / tileMap.gridWidth);

var ly = parseInt(posY / tileMap.gridHeight);

return [lx, ly];

},

getTileMapPosByPoint: function (pos) {

return cc.p(pos.x - this.touchPanel.x, pos.y - this.touchPanel.y);

},

checkOutScreen: function (pos) {

if (pos.x - Global.monsterWidth / 2 > 25 && pos.y - Global.monsterHeight / 2 > 5 && pos.x + Global.monsterWidth / 2 < Global.classicModeWidth \* Global.tileMapWidth + 50 && pos.y + Global.monsterHeight / 2 < Global.classicModeHeight \* Global.tileMapHeight + 5) {

return true;

} else {

return false;

}

},

checkMonsterByDirPos: function (pos, horizontal, vertical) {

var lx = parseInt((pos.x + horizontal - 43 + horizontal \* this.tileMap.gridWidth / 2) / this.tileMap.gridWidth);

var ly = parseInt((pos.y + vertical + vertical \* this.tileMap.gridHeight / 2) / this.tileMap.gridHeight);

ly = ly + 3;

if (this.tileMap.gridArr[ly][lx] == null) {

return false;

} else {

return true;

}

},

checkMonsterMove: function (gird) {

var bool1 = this.checkGrid(gird[0], gird[1] - 1);

var bool2 = this.checkGrid(gird[0] - 1, gird[1]);

var bool3 = this.checkGrid(gird[0] + 1, gird[1]);

var bool4 = this.checkGrid(gird[0], gird[1] + 1);

var bool = bool1 + bool2 + bool3 + bool4;

if (bool == 4) {

return true;

} else {

return false;

}

},

checkGrid: function (gird0, gird1) {

if (gird0 >= 0 && gird0 < Global.classicModeWidth && gird1 >= 3 && gird1 < Global.classicModeHeight - 2) {

if (this.tileMap.gridArr[gird1][gird0] != null) {

return 1;

} else {

return 0;

}

} else {

return 1;

}

},

computeCreateMonsterDownRows: function (tileMap) {

if (tileMap == null) return;

var allMonsterNum = 0;

var rows = 0;

for (var i = 3; i < tileMap.gridMaxY - 1; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

allMonsterNum++;

}

}

}

if (allMonsterNum > 32) {

rows = 0;

} else {

if (allMonsterNum >= 21 && allMonsterNum < 28) {

rows = 0;

}

if (allMonsterNum >= 14 && allMonsterNum < 21) {

rows = 1;

}

if (allMonsterNum >= 7 && allMonsterNum < 14) {

rows = 1;

}

if (allMonsterNum < 7) {

rows = 1;

}

}

var cols = this.searchColsOfTileMap(tileMap);

return [rows, cols];

},

searchIsFallDown: function (tileMap) {

var times = 0;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = tileMap.gridArr[i][j];

if (monster != null) {

if (monster.isFalling) {

times++;

}

}

}

}

if (times == 0) {

return true;

} else {

return false;

}

},

searchIsShakeMonster: function (tileMap) {

var times = 0;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = tileMap.gridArr[i][j];

if (monster != null) {

if (monster.isShake) {

times++;

}

}

}

}

if (times == 0) {

return true;

} else {

return false;

}

},

printArr: function () {

for (var i = Global.classicModeHeight - 3; i > 2; i--) {

var temp = "";

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null) {

switch (monster.skin) {

case 0:

temp = temp + "" + "蓝";

break;

case 1:

temp = temp + "" + "绿";

break;

case 2:

temp = temp + "" + "紫";

break;

case 3:

temp = temp + "" + "红";

break;

case 4:

temp = temp + "" + "黄";

break;

}

} else {

temp = temp + "" + 0;

}

}

}

},

starlightMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

for (var i = -1; i < 2; i++) {

for (var j = -1; j < 2; j++) {

var tempMonster = this.tileMap.gridArr[colRow[0] + i][colRow[1] + j];

if (colRow[0] + i >= 3 && colRow[0] + i < Global.classicModeHeight - 2 && colRow[1] + j >= 0 && colRow[1] + j < Global.classicModeWidth && tempMonster != null) {

if (i == 0 && j == 0) {} else {

if (!tempMonster.isShake && !tempMonster.isWaiteDie && !tempMonster.isDown && !tempMonster.isUp) {

playEffect(22);

tempMonster.initDieEffectsType(1);

if (tempMonster.effectsType == 1) {

tempMonster.flameMonsterDie();

} else {

tempMonster.playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(tempMonster);

}

}

}

}

}

},

flameMonsterFall: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

if (monster.row == 3) {

monster.playDieAni();

} else {

if (this.tileMap.gridArr[monster.row - 1][colRow[1]] != null && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isShake && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isWaiteDie && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isDown && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isUp) {

playEffect(22);

this.tileMap.gridArr[monster.row - 1][colRow[1]].initDieEffectsType(6);

if (this.tileMap.gridArr[monster.row - 1][colRow[1]].effectsType == 1) {

this.tileMap.gridArr[monster.row - 1][colRow[1]].flameMonsterDie();

} else {

this.tileMap.gridArr[monster.row - 1][colRow[1]].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[monster.row - 1][colRow[1]]);

}

var moveAction = cc.moveBy(0.03, cc.p(0, -90));

var self = this;

var callBackAction = cc.callFunc(function () {

monster.row = monster.row - 1;

self.flameMonsterFall(monster);

});

monster.runAction(cc.sequence(cc.delayTime(0.05), moveAction, callBackAction));

}

},

flameMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

var row = monster.row;

if (row == 3) {

cc.log("移到底炸周边六个");

if (colRow[1] - 1 >= 0 && colRow[1] - 1 < Global.classicModeWidth && this.tileMap.gridArr[row][colRow[1] - 1] != null) {

if (!this.tileMap.gridArr[row][colRow[1] - 1].isWaiteDie && !this.tileMap.gridArr[row][colRow[1] - 1].isShake && !this.tileMap.gridArr[row][colRow[1] - 1].isDown && !this.tileMap.gridArr[row][colRow[1] - 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row][colRow[1] - 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row][colRow[1] - 1].effectsType == 1) {

this.tileMap.gridArr[row][colRow[1] - 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row][colRow[1] - 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1] - 1]);

}

}

if (colRow[1] - 1 >= 0 && colRow[1] - 1 < Global.classicModeWidth && row + 1 >= 3 && row + 1 < Global.classicModeHeight - 2 && this.tileMap.gridArr[row + 1][colRow[1] - 1] != null) {

if (!this.tileMap.gridArr[row + 1][colRow[1] - 1].isWaiteDie && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isShake && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isDown && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row + 1][colRow[1] - 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row + 1][colRow[1] - 1].effectsType == 1) {

this.tileMap.gridArr[row + 1][colRow[1] - 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row + 1][colRow[1] - 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row + 1][colRow[1] - 1]);

}

}

if (colRow[1] + 1 >= 0 && colRow[1] + 1 < Global.classicModeWidth && this.tileMap.gridArr[row][colRow[1] + 1] != null) {

if (!this.tileMap.gridArr[row][colRow[1] + 1].isWaiteDie && !this.tileMap.gridArr[row][colRow[1] + 1].isShake && !this.tileMap.gridArr[row][colRow[1] + 1].isDown && !this.tileMap.gridArr[row][colRow[1] + 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row][colRow[1] + 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row][colRow[1] + 1].effectsType == 1) {

this.tileMap.gridArr[row][colRow[1] + 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row][colRow[1] + 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1] + 1]);

}

}

if (colRow[1] + 1 >= 0 && colRow[1] + 1 < Global.classicModeWidth && row + 1 >= 3 && row + 1 < Global.classicModeHeight - 2 && this.tileMap.gridArr[row + 1][colRow[1] + 1] != null) {

if (!this.tileMap.gridArr[row + 1][colRow[1] + 1].isWaiteDie && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isShake && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isDown && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row + 1][colRow[1] + 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row + 1][colRow[1] + 1].effectsType == 1) {

this.tileMap.gridArr[row + 1][colRow[1] + 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row + 1][colRow[1] + 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row + 1][colRow[1] + 1]);

}

}

}

},

lightningMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

for (var i = 0; i < Global.classicModeWidth; i++) {

if (i != colRow[1] && this.tileMap.gridArr[colRow[0]][i] != null) {

if (!this.tileMap.gridArr[colRow[0]][i].isWaiteDie && !this.tileMap.gridArr[colRow[0]][i].isShake && !this.tileMap.gridArr[colRow[0]][i].isDown && !this.tileMap.gridArr[colRow[0]][i].isUp) {

playEffect(22);

this.tileMap.gridArr[colRow[0]][i].initDieEffectsType(5);

if (this.tileMap.gridArr[colRow[0]][i].effectsType == 1) {

this.tileMap.gridArr[colRow[0]][i].flameMonsterDie();

} else {

this.tileMap.gridArr[colRow[0]][i].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[colRow[0]][i]);

}

}

}

for (var j = 0; j < 9; j++) {

if (j + 3 != colRow[0] && this.tileMap.gridArr[j + 3][colRow[1]] != null) {

if (!this.tileMap.gridArr[j + 3][colRow[1]].isWaiteDie && !this.tileMap.gridArr[j + 3][colRow[1]].isShake && !this.tileMap.gridArr[j + 3][colRow[1]].isDown && !this.tileMap.gridArr[j + 3][colRow[1]].isUp) {

playEffect(22);

this.tileMap.gridArr[j + 3][colRow[1]].initDieEffectsType(5);

if (this.tileMap.gridArr[j + 3][colRow[1]].effectsType == 1) {

this.tileMap.gridArr[j + 3][colRow[1]].flameMonsterDie();

} else {

this.tileMap.gridArr[j + 3][colRow[1]].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[j + 3][colRow[1]]);

}

}

}

},

blackHoleMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

var self = this;

var monsterArr = [];

for (var j = colRow[0] - 2; j < colRow[0] + 3; j++) {

for (var i = colRow[1] - 2; i < colRow[1] + 3; i++) {

if (j >= 3 && j < Global.classicModeHeight - 2 && i >= 0 && i < Global.classicModeWidth && !(i == colRow[1] && j == colRow[0])) {

var rotaMonster = this.tileMap.gridArr[j][i];

if (rotaMonster != null && rotaMonster.isBox == false && rotaMonster.isAdvancedBox == false && rotaMonster.isBoxMonster == false && rotaMonster.isAdvancedBoxMonster == false && !rotaMonster.isWaiteDie && !rotaMonster.isShake && !rotaMonster.isDown && !rotaMonster.isUp) {

cc.log("//播放吸入动画");

rotaMonster.isWaiteDie = true;

var pStart = rotaMonster.getPosition();

var pEnd = monster.getPosition();

rotaMonster.setMonsterAngle(pStart, pEnd);

rotaMonster.addScheduleTime();

rotaMonster.setMonsterData(pStart, pEnd);

monsterArr.push(rotaMonster);

}

}

}

}

if (monsterArr.length != 0) {

this.gameControl.createRemoveCycloneMonsterList(monsterArr);

}

},

useDyeingBottle: function (grid) {

var skin = randomNums(0, 5);

for (var i = -1; i < 2; i++) {

for (var j = -1; j < 2; j++) {

if (grid[0] + j >= 0 && grid[0] + j < Global.classicModeWidth && grid[1] + i >= 3 && grid[1] + i < Global.classicModeHeight - 2 && this.tileMap.gridArr[grid[1] + i][grid[0] + j] != null && (this.tileMap.gridArr[grid[1] + i][grid[0] + j].isCageMonster || this.tileMap.gridArr[grid[1] + i][grid[0] + j].isBoxMonster || this.tileMap.gridArr[grid[1] + i][grid[0] + j].isAdvancedBoxMonster || this.tileMap.gridArr[grid[1] + i][grid[0] + j].isMonster || this.tileMap.gridArr[grid[1] + i][grid[0] + j].isMonsterKing)) {

var monster = this.tileMap.gridArr[grid[1] + i][grid[0] + j];

if (!monster.isShake && !monster.isDown && !monster.isUp) {

if (monster.isMonsterKing) {

monster.monsterKing.setSkin(Global.monsterKingSkin[skin]);

monster.skin = skin;

monster.setMonsterKingLv();

} else {

monster.monster.setSkin(Global.monsterSkin[skin]);

monster.skin = skin;

}

}

}

}

}

gamePlayerData.money = gamePlayerData.money - 500;

currentScene.gameLayer.refreshCurrentMoney();

currentScene.gameLayer.isDyeing = false;

currentScene.gameLayer.isSelectDyeBottle = false;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = false;

this.fallMonstersCallFun();

},

useToiletPlug: function (grid) {

cc.log("useToiletPlug");

if (this.tileMap.gridArr[grid[1]][grid[0]] != null) {

var skin = this.tileMap.gridArr[grid[1]][grid[0]].skin;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null) {

if (monster.skin != null && monster.skin == skin) {

monster.initDieEffectsType(1);

if (monster.effectsType == 1) {

monster.flameMonsterDie();

} else {

monster.playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(monster);

}

}

}

}

}

gamePlayerData.money = gamePlayerData.money - 200;

currentScene.gameLayer.refreshCurrentMoney();

currentScene.gameLayer.isPlayToilet = false;

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

},

removedFromBox: function (monster, type, pos) {

monster.monster.removeFromParent();

monster.monster = null;

var ID = monster.skin + 1;

monster.skin = -1;

if (type == 1) {

monster.isBoxMonster = false;

monster.isBox = true;

}

if (type == 2) {

monster.isAdvancedBoxMonster = false;

monster.isAdvancedBox = true;

}

var tempMonster = new SpecialMonsterNode(ID);

if (tempMonster != null) {

tempMonster.x = pos.x;

tempMonster.y = pos.y;

this.touchPanel.addChild(tempMonster);

}

return tempMonster;

},

updateCheckout: function (dt) {

if (!this.isGameOver && !this.isGameWin) {

if (this.countTime >= 0) {

if (this.checkIsFullMonster()) {

if (!this.isFullMonster) {

this.isFullMonster = true;

this.checkGameLose();

this.stopScheduleCheckout();

}

} else {

this.checkFallTooth(this.countTime);

this.checkShowGameAlarm();

this.checkAddTimeMonster();

if (this.score >= 1000 && this.score < 3000 && !this.isShowTipsTwo) {

this.showGameTipsByID(1);

}

if (this.score >= 3000 && !this.isShowTipsThree) {

this.showGameTipsByID(2);

}

this.countTime--;

this.refreshTime();

this.checkGameWin();

}

} else {

this.isTimeUP = true;

this.checkGameLose();

this.stopScheduleCheckout();

}

}

},

refreshTime: function () {

var second = this.countTime % 60;

second = second >= 10 ? second : "0" + second;

var minute = parseInt(this.countTime / 60);

minute = minute >= 10 ? minute : "0" + minute;

var str = minute + ":" + second;

this.rootNode.BitmapFontLabel\_6.setString(str);

},

checkGameWin: function () {

var firstTargetCondition = false;

var secondTargetCondition = false;

if (this.gameTarget[0] == 11) {

if (this.score >= this.gameTargetScore) {

firstTargetCondition = true;

if (!this.isPlayCompleteTargetOneEffect) {

this.isPlayCompleteTargetOneEffect = true;

playEffect(27);

}

}

cc.log("第一个目标是分数");

if (this.gameTarget[2] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[2] - 1];

if (tempnum >= this.gameTarget[3]) {

if (!this.isPlayCompleteTargetTwoEffect) {

this.isPlayCompleteTargetTwoEffect = true;

playEffect(27);

}

secondTargetCondition = true;

}

} else {

secondTargetCondition = true;

}

} else {

if (this.gameTarget[0] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[0] - 1];

if (tempnum >= this.gameTarget[1]) {

firstTargetCondition = true;

if (!this.isPlayCompleteTargetOneEffect) {

this.isPlayCompleteTargetOneEffect = true;

playEffect(27);

}

}

if (this.gameTarget[2] != null) {

var tempnum = currentScene.gameRecordRemoveMonster[this.gameTarget[2] - 1];

var str = tempnum + "/" + this.gameTarget[3];

this.rootNode.BitmapFontLabel\_5\_0.setString(str);

if (tempnum >= this.gameTarget[3]) {

secondTargetCondition = true;

if (!this.isPlayCompleteTargetTwoEffect) {

this.isPlayCompleteTargetTwoEffect = true;

playEffect(27);

}

}

} else {

secondTargetCondition = true;

}

} else {

secondTargetCondition = true;

}

}

if (firstTargetCondition && secondTargetCondition) {

cc.log("弹胜利界面");

this.isGameWin = true;

var score = this.score;

var starNum = this.calculateGetStar();

if (gamePlayerData.levelStarData[Global.currentSelectLevel - 1] != null) {

if (starNum >= gamePlayerData.levelStarData[Global.currentSelectLevel - 1]) {

gamePlayerData.levelStarData[Global.currentSelectLevel - 1] = starNum;

}

} else {

gamePlayerData.levelStarData[Global.currentSelectLevel - 1] = starNum;

}

if (Global.currentSelectLevel == gamePlayerData.currentLevel) {

if (gamePlayerData.currentLevel == Global.maxSelectLevel) {

Global.currentSelectLevel = gamePlayerData.currentLevel;

} else {

gamePlayerData.currentLevel++;

Global.currentSelectLevel = gamePlayerData.currentLevel;

}

} else {

if (Global.currentSelectLevel < gamePlayerData.currentLevel) {

Global.currentSelectLevel++;

}

}

currentScene.gameLayer.boomOneByOne();

var callFunAction = cc.callFunc(function () {

currentScene.gameLayer.showGameAlarm(false);

currentScene.gameWin(score, starNum);

});

this.runAction(cc.sequence(cc.delayTime(1), callFunAction));

}

savePlayerData();

},

checkGameLose: function () {

cc.log("弹失败界面");

this.isGameOver = true;

var callFunAction = cc.callFunc(function () {

currentScene.gameLayer.showGameAlarm(false);

currentScene.gameLose();

});

var callFunActionI = cc.callFunc(function () {

currentScene.gameLayer.playTimeUPAnimation();

});

if (currentScene.gameLayer.isTimeUP) {

currentScene.gameLayer.isTimeUP = false;

this.runAction(cc.sequence(callFunActionI, cc.delayTime(1), callFunAction));

} else {

this.runAction(cc.sequence(cc.delayTime(1), callFunAction));

}

},

calculateGetStar: function () {

if (this.countTime >= this.gameStarThree) {

return 3;

} else if (this.countTime >= this.gameStarTwo && this.countTime < this.gameStarThree) {

return 2;

} else if (this.countTime >= this.gameStarOne && this.countTime < this.gameStarTwo) {

return 1;

} else {

cc.log("没有得星");

return 0;

}

},

showGameAlarm: function (boolean) {

if (boolean) {

this.rootNode.GameAlarm\_1.visible = true;

this.rootNode.GameAlarm\_1.runAction(cc.sequence(cc.fadeIn(0.2), cc.fadeOut(0.5)).repeatForever());

if (!this.isPlayWarningEffect) {

this.isPlayWarningEffect = true;

playWarningEffect(25);

}

} else {

this.rootNode.GameAlarm\_1.visible = false;

if (this.isPlayWarningEffect) {

this.isPlayWarningEffect = false;

stopWarningEffect();

}

}

},

checkShowGameAlarm: function () {

var monsterCount = 0;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null && !monster.isShake) {

monsterCount++;

}

}

}

if (this.countTime <= 15 || monsterCount > 45) {

if (this.countTime == 15) {

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 4);

}

if (this.countTime == 10) {

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 3);

}

if (this.countTime == 5) {

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 2);

}

if (!this.isShowGameAlarm) {

this.isShowGameAlarm = true;

this.showGameAlarm(this.isShowGameAlarm);

}

} else {

if (this.isShowGameAlarm) {

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 10);

}

this.isShowGameAlarm = false;

this.showGameAlarm(this.isShowGameAlarm);

}

},

addToothAnimation: function (type, pos) {

var sprite = new cc.Sprite("#tooth" + type + "0001" + ".png");

var frames = [];

var picNum = 0;

if (type == 1) {

picNum = 16;

}

if (type == 2) {

picNum = 19;

}

for (var i = 1; i <= picNum; i++) {

var frame = cc.spriteFrameCache.getSpriteFrame("tooth" + type + "000" + i + ".png");

frames.push(frame);

}

var Animation = new cc.Animation.create(frames, 0.04);

var animate = new cc.Animate(Animation);

var callFun = cc.callFunc(function () {

sprite.removeFromParent();

});

sprite.runAction(cc.sequence(animate, callFun));

sprite.setPosition(cc.p(pos.x - 2, pos.y - 10));

this.addChild(sprite, 5);

},

checkFallTooth: function (count) {

for (var i = 0; i < 12; i++) {

if (count == this.fallToothTime[i]) {

var ID = 11 - i;

var tooth = this.rootNode.getChildByName("GameTooth" + ID);

tooth.visible = false;

var pos = tooth.getPosition();

var type = 2;

if (ID == 2 || ID == 5 || ID == 8) {

type = 1;

}

this.addToothAnimation(type, pos);

}

}

},

calculateFallToothTime: function () {

this.fallToothTime = [];

var timeOne = parseInt((this.countTime - this.gameStarThree) / 4);

this.fallToothTime.push(this.countTime - timeOne);

this.fallToothTime.push(this.countTime - 2 \* timeOne);

this.fallToothTime.push(this.countTime - 3 \* timeOne);

this.fallToothTime.push(this.gameStarThree);

var timeTwo = parseInt((this.gameStarThree - this.gameStarTwo) / 3);

this.fallToothTime.push(this.gameStarThree - timeTwo);

this.fallToothTime.push(this.gameStarThree - 2 \* timeTwo);

this.fallToothTime.push(this.gameStarTwo);

var timeThree = parseInt((this.gameStarTwo - this.gameStarOne) / 3);

this.fallToothTime.push(this.gameStarTwo - timeThree);

this.fallToothTime.push(this.gameStarTwo - 2 \* timeThree);

this.fallToothTime.push(this.gameStarOne);

var timeFour = parseInt(this.gameStarOne / 2);

this.fallToothTime.push(this.gameStarOne - timeFour);

this.fallToothTime.push(1);

},

checkAddTimeMonster: function () {

var timeMonsterID = parseInt(this.score / 5000);

if (timeMonsterID != this.addTimeMonsterID && timeMonsterID > this.addTimeMonsterID) {

this.addTimeMonsterID = timeMonsterID;

this.isCanAddTimeMonster = true;

}

},

checkIsFullMonster: function () {

var monsterCount = 0;

for (var i = 3; i < Global.classicModeHeight - 2; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null && !monster.isShake) {

monsterCount++;

}

}

}

if (monsterCount == 63) {

return true;

} else {

return false;

}

},

playTimeUPAnimation: function () {

var timeUP = new cc.Sprite(res.timeUP\_png);

timeUP.x = 360;

timeUP.y = 640;

this.addChild(timeUP, 15);

timeUP.setScale(0.1, 0.1);

var callFun = cc.callFunc(function () {

timeUP.removeFromParent();

});

timeUP.runAction(cc.sequence(cc.scaleTo(0.2, 1.15, 1.15), cc.scaleTo(0.2, 0.95, 0.95), cc.scaleTo(0.2, 1, 1), cc.delayTime(0.2), callFun));

},

showGameTipsByID: function (ID) {

if (Global.currentMode == 1) {

if (Global.currentSelectLevel == 1) {

switch (ID) {

case 1:

if (!currentScene.gameLayer.isShowTipsTwo) {

currentScene.gameLayer.isShowTipsTwo = true;

var callFun = cc.callFunc(function () {

currentScene.gameTips(2);

});

currentScene.runAction(cc.sequence(cc.delayTime(0.1), callFun));

}

break;

case 2:

if (this.effectsType != -1) {

if (Global.currentMode == 1) {

if (Global.currentSelectLevel == 1) {

if (!currentScene.gameLayer.isShowTipsThree) {

currentScene.gameLayer.isShowTipsThree = true;

var callFun = cc.callFunc(function () {

currentScene.gameTips(3);

});

currentScene.runAction(cc.sequence(cc.delayTime(0.1), callFun));

}

}

}

}

break;

}

}

}

},

boomOneByOne: function () {

var isExistMonster = false;

for (var i = Global.classicModeHeight - 2; i >= 3; i--) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null && !monster.isUp && !monster.isDown && !monster.isWaiteDie && monster.isMonster && monster.effectsType == -1 && !monster.isShake) {

this.monsterPlayBoom(monster);

isExistMonster = true;

break;

}

}

if (isExistMonster) {

break;

}

}

},

monsterPlayBoom: function (monster) {

monster.isWaiteDie = true;

monster.visible = false;

var sprite = this.createBoomEffects(monster, monster.skin + 2);

sprite.x = currentScene.gameLayer.tileMapPos[monster.rowcol[0]][monster.rowcol[1]].x + 3;

sprite.y = currentScene.gameLayer.tileMapPos[monster.rowcol[0]][monster.rowcol[1]].y + 180;

currentScene.addChild(sprite, 5);

},

createBoomEffects: function (monster, effectsType) {

var arr = [];

arr = Global.monsterDieEfects[effectsType];

var sprite = new cc.Sprite("#" + arr[0] + "0000" + arr[1] + ".png");

var frames = [];

for (var i = arr[1]; i <= arr[2]; i++) {

var frame = cc.spriteFrameCache.getSpriteFrame(arr[0] + "0000" + i + ".png");

frames.push(frame);

}

var Animation = new cc.Animation.create(frames, 0.07);

var animate = new cc.Animate(Animation);

var self = sprite;

var myself = monster;

var myLayer = this;

var callFunAction = cc.callFunc(function () {

self.removeFromParent();

myLayer.boomOneByOne();

});

var callAction = cc.callFunc(function () {

if (myself != null && myself.monster != null) {

myself.removeMonster();

}

});

sprite.runAction(cc.sequence(animate, callAction, callFunAction));

return sprite;

}

});

var gameTargetLayerTargetOffset = 100;

var GameTargetLayer = cc.Layer.extend({

rootNode: null,

gameTarget: null,

countTime: 0,

score: 0,

gameStarOne: 0,

gameStarTwo: 0,

gameStarThree: 0,

gameTargetScore: 0,

gameTargetOneID: 0,

gameTargetOneNum: 0,

gameTargetTwoID: 0,

gameTargetTwoNum: 0,

gameInitFallCD: 0,

gameFallCD: 0,

ctor: function () {

this.\_super();

this.init();

},

init: function () {

this.initGameTargetData();

this.initUI();

},

initUI: function () {

this.rootNode = loadCSS(res.gameTargetLayer\_json);

this.addChild(this.rootNode);

this.initGameTargetUI();

this.rootNode.Image\_4.visible = false;

this.rootNode.Image\_4.setScale(0.5, 0.5);

this.rootNode.Image\_4.runAction(cc.fadeOut(0.1));

var self = this;

self.setCascadeOpacityEnabled(true);

var callFunAction = cc.callFunc(function () {

self.rootNode.Image\_1.visible = false;

self.rootNode.Image\_4.visible = true;

playEffect(24);

self.rootNode.Image\_4.runAction(cc.spawn(cc.scaleTo(0.2, 1.2, 1.2), cc.fadeIn(0.3), cc.scaleTo(0.1, 1.0, 1.0)));

});

var callFunActionII = cc.callFunc(function () {

self.removeFromParent();

var picID = -1;

switch (Global.currentSelectLevel) {

case 1:

picID = 1;

break;

case 2:

picID = 4;

break;

case 3:

picID = 5;

break;

case 4:

picID = 6;

break;

case 5:

picID = 7;

break;

}

if (picID != -1) {

currentScene.gameTips(picID);

}

currentScene.gameLayer.scheduleFun();

});

this.rootNode.Image\_1.runAction(cc.sequence(cc.moveTo(0.3, cc.p(360, 640)), cc.delayTime(1.5), callFunAction, cc.delayTime(1), callFunActionII));

},

initGameTargetData: function () {

this.gameTarget = [];

this.countTime = Global.gameTargetData[Global.currentSelectLevel - 1][0];

this.gameStarOne = Global.gameTargetData[Global.currentSelectLevel - 1][1];

this.gameStarTwo = Global.gameTargetData[Global.currentSelectLevel - 1][2];

this.gameStarThree = Global.gameTargetData[Global.currentSelectLevel - 1][3];

this.gameTargetScore = Global.gameTargetData[Global.currentSelectLevel - 1][4];

this.gameTargetOneID = Global.gameTargetData[Global.currentSelectLevel - 1][5];

this.gameTargetOneNum = Global.gameTargetData[Global.currentSelectLevel - 1][6];

this.gameTargetTwoID = Global.gameTargetData[Global.currentSelectLevel - 1][7];

this.gameTargetTwoNum = Global.gameTargetData[Global.currentSelectLevel - 1][8];

this.gameInitFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][9];

this.gameFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][10];

},

initGameTargetUI: function () {

if (this.gameTargetScore > 0) {

this.gameTarget.push(11);

this.gameTarget.push(this.gameTargetScore);

this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[10]);

this.rootNode.Image\_1.BitmapFontLabel\_2.setString(this.gameTargetScore + "");

if (this.gameTargetOneID > 0) {

this.gameTarget.push(this.gameTargetOneID);

this.gameTarget.push(this.gameTargetOneNum);

this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID - 1]);

this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x" + this.gameTargetOneNum);

} else {

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x" + this.gameTargetTwoNum);

} else {

this.rootNode.Image\_1.Sprite\_3.visible = false;

this.rootNode.Image\_1.BitmapFontLabel\_3.visible = false;

this.rootNode.Image\_1.Sprite\_2.x += gameTargetLayerTargetOffset;

this.rootNode.Image\_1.BitmapFontLabel\_2.x += gameTargetLayerTargetOffset;

}

}

} else {

if (this.gameTargetOneID > 0) {

this.gameTarget.push(this.gameTargetOneID);

this.gameTarget.push(this.gameTargetOneNum);

this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID - 1]);

this.rootNode.Image\_1.BitmapFontLabel\_2.setString("x" + this.gameTargetOneNum);

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x" + this.gameTargetTwoNum);

} else {

this.rootNode.Image\_1.Sprite\_3.visible = false;

this.rootNode.Image\_1.BitmapFontLabel\_3.visible = false;

this.rootNode.Image\_1.Sprite\_2.x += gameTargetLayerTargetOffset;

this.rootNode.Image\_1.BitmapFontLabel\_2.x += gameTargetLayerTargetOffset;

}

} else {

if (this.gameTargetTwoID > 0) {

this.gameTarget.push(this.gameTargetTwoID);

this.gameTarget.push(this.gameTargetTwoNum);

this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID - 1]);

this.rootNode.Image\_1.BitmapFontLabel\_2.setString("x" + this.gameTargetTwoNum);

this.rootNode.Image\_1.Sprite\_3.visible = false;

this.rootNode.Image\_1.BitmapFontLabel\_3.visible = false;

this.rootNode.Image\_1.Sprite\_2.x += gameTargetLayerTargetOffset;

this.rootNode.Image\_1.BitmapFontLabel\_2.x += gameTargetLayerTargetOffset;

} else {

this.rootNode.Image\_1.Sprite\_2.visible = false;

this.rootNode.Image\_1.BitmapFontLabel\_2.visible = false;

}

}

}

}

});

var GamePauseLayer = cc.Layer.extend({

rootNode: null,

home: null,

music: null,

musicoff: null,

help: null,

continue: null,

onEnter: function () {

this.\_super();

this.init();

},

init: function () {

this.initUI();

this.initText();

this.initListener();

},

initText: function () {},

initUI: function () {

this.rootNode = loadCSS(res.GamePauseLayer\_json);

this.addChild(this.rootNode);

this.home = this.rootNode.Image\_2.Button\_1;

this.home.tag = 1;

this.home.setPressedActionEnabled(true);

this.music = this.rootNode.Image\_2.Button\_4;

this.music.tag = 2;

this.music.setPressedActionEnabled(true);

this.musicoff = this.rootNode.Image\_2.Button\_5;

this.musicoff.tag = 5;

this.musicoff.setPressedActionEnabled(true);

this.help = this.rootNode.Image\_2.Button\_2;

this.help.tag = 3;

this.help.setPressedActionEnabled(true);

this.continue = this.rootNode.Image\_2.Button\_3;

this.continue.tag = 4;

this.continue.setPressedActionEnabled(true);

this.initMusic();

},

initMusic: function () {

if (gamePlayerData.soundSwitch) {

this.music.visible = true;

this.musicoff.visible = false;

} else {

this.music.visible = false;

this.musicoff.visible = true;

}

},

initListener: function () {

this.home.addTouchEventListener(this.btnCallFun);

this.music.addTouchEventListener(this.btnCallFun);

this.help.addTouchEventListener(this.btnCallFun);

this.continue.addTouchEventListener(this.btnCallFun);

this.musicoff.addTouchEventListener(this.btnCallFun);

},

btnCallFun: function (sender, type) {

switch (sender.tag) {

case 1:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("home");

if (cc.director.isPaused()) {

cc.director.resume();

currentScene.gameLayer.isPause = false;

}

if (Global.currentMode == 1) {

transitions(SelectScene);

}

if (Global.currentMode == 2) {

transitions(LoginScene);

}

break;

}

break;

case 2:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("music");

gamePlayerData.soundSwitch = false;

stopBgMusic(1);

currentScene.gamePauseLayer.initMusic();

break;

}

break;

case 3:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("help");

if (cc.director.isPaused()) {

cc.director.resume();

currentScene.gameLayer.isPause = false;

}

transitions(GamePlayScene);

break;

}

break;

case 4:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("continue");

if (currentScene.gamePauseLayer != null) {

if (cc.director.isPaused()) {

cc.director.resume();

currentScene.gameLayer.isPause = false;

}

currentScene.gamePauseLayer.removeFromParent();

currentScene.gamePauseLayer = null;

}

break;

}

break;

case 5:

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("musicoff");

gamePlayerData.soundSwitch = true;

playBgMusic(1);

currentScene.gamePauseLayer.initMusic();

break;

}

break;

}

savePlayerData();

}

});

var EndlessGameLayer = cc.Layer.extend({

rootNode: null,

btnPause: null,

btnAddGold: null,

btnToiletSucker: null,

btnDyeBottle: null,

panel: null,

touchPanel: null,

tileMap: null,

tileMapx: null,

tileMapPos: null,

tileMapUp: null,

tileMapDown: null,

tempMonsterArr: null,

tempMonsterArrs: null,

removeRows: null,

touchMonster: null,

isCreateMonsterDown: false,

isCreateCompleted: false,

tempColsArr: null,

isClearMonster: false,

isTouchDown: false,

touchPosition: null,

gameControl: null,

touchPosInBox: null,

frontGrid: null,

currentGrid: null,

isGenerateMonster: false,

connectMonsterArrs: null,

isTouchInBox: false,

isSelectDyeBottle: false,

isDyeing: false,

isSelectToiletSucker: false,

isPlayToilet: false,

isBanFall: false,

countTime: 0,

score: 0,

gameStarOne: 0,

gameStarTwo: 0,

gameStarThree: 0,

gameTargetScore: 0,

gameTargetOneID: 0,

gameTargetOneNum: 0,

gameTargetTwoID: 0,

gameTargetTwoNum: 0,

gameInitFallCD: 0,

gameFallCD: 0,

gameTarget: null,

isGameWin: false,

isGameOver: false,

isTouchOnMouseUp: false,

LV: 1,

isShowGameAlarm: false,

scoreBar: null,

gameLevelUpNode: null,

ctor: function () {

this.\_super();

this.init();

this.scheduleOnce(this.scheduleFallMonster, 10);

},

scheduleFallMonster: function () {

this.createMonsterByRows(1, 1);

this.schedule(this.fallMonster, 10);

},

onEnter: function () {

this.\_super();

this.schedule(this.updateCheckout, 1);

},

init: function () {

this.initUI();

this.initData();

this.initListener();

},

initGameTargetData: function () {

this.gameTarget = [];

this.countTime = Global.gameTargetData[Global.currentSelectLevel - 1][0];

this.gameStarOne = Global.gameTargetData[Global.currentSelectLevel - 1][1];

this.gameStarTwo = Global.gameTargetData[Global.currentSelectLevel - 1][2];

this.gameStarThree = Global.gameTargetData[Global.currentSelectLevel - 1][3];

this.gameTargetScore = Global.gameTargetData[Global.currentSelectLevel - 1][4];

this.gameTargetOneID = Global.gameTargetData[Global.currentSelectLevel - 1][5];

this.gameTargetOneNum = Global.gameTargetData[Global.currentSelectLevel - 1][6];

this.gameTargetTwoID = Global.gameTargetData[Global.currentSelectLevel - 1][7];

this.gameTargetTwoNum = Global.gameTargetData[Global.currentSelectLevel - 1][8];

this.gameInitFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][9];

this.gameFallCD = Global.gameTargetData[Global.currentSelectLevel - 1][10];

},

initData: function () {

this.tempMonsterArrs = [];

this.removeRows = [];

this.connectMonsterArrs = [];

this.tempColsArr = [0, 0, 0, 0, 0, 0, 0];

this.tileMap = new TileMap();

var tileMapData = this.createTileMapData(Global.classicModeOffsetX, Global.classicModeOffsetY, Global.tileMapWidth, Global.tileMapHeight, Global.classicModeWidth, Global.classicModeHeight, null);

this.tileMap.initialize(tileMapData);

this.createMapPoint();

},

createMapPoint: function () {

this.tileMapPos = [];

for (var row = 0; row < Global.classicModeHeight; row++) {

var arr = [];

for (var col = 0; col < Global.classicModeWidth; col++) {

if (row == Global.classicModeHeight - 1) {

var point = cc.p(col \* Global.tileMapWidth + Global.monsterWidth / 2 + 43, Global.tileMapHeight \* (row - 3) + Global.monsterHeight / 2 + 30);

} else {

var point = cc.p(col \* Global.tileMapWidth + Global.monsterWidth / 2 + 43, Global.tileMapHeight \* (row - 3) + Global.monsterHeight / 2);

}

arr.push(point);

}

this.tileMapPos.push(arr);

}

},

createTileMapData: function (offsetX, offsetY, gridWidth, gridHeight, gridMaxX, gridMaxY, defaultGridValue) {

var data = new Object();

data.offsetX = offsetX;

data.offsetY = offsetY;

data.gridWidth = gridWidth;

data.gridHeight = gridHeight;

data.gridMaxX = gridMaxX;

data.gridMaxY = gridMaxY;

data.defaultGridValue = defaultGridValue;

return data;

},

initUI: function () {

this.rootNode = loadCSS(res.EndlessGameLayer\_json);

this.addChild(this.rootNode);

this.btnPause = this.rootNode.Button\_1;

this.btnAddGold = this.rootNode.Image\_8.Button\_2;

this.btnAddGold.setPressedActionEnabled(true);

this.btnDyeBottle = this.rootNode.Image\_9;

this.rootNode.dyebottlex\_1.visible = false;

this.btnToiletSucker = this.rootNode.Image\_12;

this.rootNode.toiletsuckerx\_2.visible = false;

this.rootNode.Image\_1.setLocalZOrder(-3);

this.rootNode.GameTongue\_1.setLocalZOrder(-2);

this.touchPanel = new cc.Layer();

this.touchPanel.width = 720;

this.touchPanel.height = 960;

this.touchPanel.x = 3;

this.touchPanel.y = 180;

this.rootNode.addChild(this.touchPanel, -1);

this.gameControl = currentScene.gameControl;

this.rootNode.BitmapFontLabel\_1.setString("" + 1);

this.rootNode.BitmapFontLabel\_2.setString("" + 0);

this.rootNode.GameAlarm\_1.visible = false;

this.refreshCurrentMoney();

this.scoreBar = this.rootNode.Image\_14.LoadingBar\_2;

this.scoreBar.setPercent(0);

this.refreshBtnAddMoney();

this.gameLevelUpNode = loadCSS(res.GameLevelUpNode\_json);

this.gameLevelUpNode.x = 360;

this.gameLevelUpNode.y = 1920;

this.addChild(this.gameLevelUpNode, 20);

this.showGameLevelUpNode();

},

refreshBtnAddMoney: function () {

if (gamePlayerData.isJiHuo == 0) {

this.btnAddGold.visible = false;

}

if (gamePlayerData.isJiHuo == 1) {

this.btnAddGold.visible = true;

}

},

refreshCurrentLV: function () {

this.rootNode.BitmapFontLabel\_1.setString("" + this.LV);

},

refreshCurrentScore: function (recordConnectRemoveTimes, removeCount) {

if (!this.isGameOver) {

var addScore = (recordConnectRemoveTimes \* 100 + 300) \* (removeCount - 2);

this.score += addScore;

this.rootNode.BitmapFontLabel\_2.setString(this.score + "");

currentScene.createEncourage(addScore);

}

this.updateLVAndTime(this.score);

},

refreshCurrentMoney: function () {

this.rootNode.Image\_8.BitmapFontLabel\_3.setString("" + gamePlayerData.money);

},

updateLVAndTime: function (score) {

var Lv = 1;

if (score < 10000) {

Lv = 1;

this.refreshScoreBar(0, 10000);

}

if (score >= 10000 && score < 25000) {

Lv = 2;

this.refreshScoreBar(10000, 25000);

} else if (score >= 25000 && score < 40000) {

Lv = 3;

this.refreshScoreBar(25000, 40000);

} else if (score >= 40000 && score < 58000) {

Lv = 4;

this.refreshScoreBar(40000, 58000);

} else if (score >= 58000 && score < 70000) {

Lv = 5;

this.refreshScoreBar(58000, 70000);

} else if (score >= 70000 && score < 95000) {

Lv = 6;

this.refreshScoreBar(70000, 95000);

} else if (score >= 95000 && score < 115000) {

Lv = 7;

this.refreshScoreBar(95000, 115000);

} else if (score >= 115000 && score < 125000) {

Lv = 8;

this.refreshScoreBar(115000, 125000);

} else if (score >= 125000 && score < 150000) {

Lv = 9;

this.refreshScoreBar(125000, 150000);

} else if (score >= 150000) {

Lv = parseInt(score - 150000) / 20000 + 10;

this.refreshScoreBar(150000 + (Lv - 10) \* 20000, 150000 + (Lv - 10) \* 20000 + 20000);

} else {

cc.log("score < 10000");

}

if (this.LV == Lv) {

cc.log("this.Lv == Lv");

} else {

this.LV = Lv;

this.refreshCurrentLV();

this.unschedule(this.fallMonster);

if (Lv <= 9) {

var fallCD = Global.fallCD[Lv - 2];

this.schedule(this.fallMonster, fallCD);

} else {

this.schedule(this.fallMonster, 3);

}

this.showGameLevelUpNode();

}

},

refreshScoreBar: function (scoreMin, scoreMax) {

var value = (this.score - scoreMin) / (scoreMax - scoreMin) \* 100;

this.scoreBar.setPercent(value);

},

refreshCurrentScoreByScore: function (score) {

this.score += score;

this.rootNode.BitmapFontLabel\_2.setString(this.score + "");

this.updateLVAndTime(this.score);

},

initListener: function () {

this.btnPause.addTouchEventListener(this.btnPauseFun, this.btnPause);

this.btnAddGold.addTouchEventListener(this.btnAddGoldFun, this.btnAddGold);

engine.gameEvent.createMouseDown(this.btnDyeBottle, this.btnDyeBottleFun);

engine.gameEvent.createMouseDown(this.btnToiletSucker, this.btnToiletSuckerFun);

this.addTouchListener();

},

btnDyeBottleFun: function () {

playEffect(2);

cc.log("btnDyeBottleFun");

if (currentScene.gameLayer.isSelectToiletSucker) return;

if (currentScene.gameLayer.isSelectDyeBottle) {

if (currentScene.gameLayer.isDyeing) {

cc.log("正在染色");

} else {

currentScene.gameLayer.isSelectDyeBottle = false;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = false;

}

} else {

if (gamePlayerData.money >= 500) {

currentScene.gameLayer.isSelectDyeBottle = true;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = true;

}

}

savePlayerData();

},

btnToiletSuckerFun: function () {

playEffect(2);

cc.log("btnToiletSuckerFun");

if (currentScene.gameLayer.isSelectDyeBottle) return;

if (currentScene.gameLayer.isSelectToiletSucker) {

if (currentScene.gameLayer.isPlayToilet) {

cc.log("正在播放马桶塞动画");

} else {

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

}

} else {

if (gamePlayerData.money >= 200) {

currentScene.gameLayer.isSelectToiletSucker = true;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = true;

}

}

savePlayerData();

},

btnPauseFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(4);

cc.log("btnPauseFun");

if (!this.isGameOver && !this.isGameWin) {

this.isPause = true;

cc.director.pause();

currentScene.gamePause();

}

break;

}

},

btnAddGoldFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnAddGoldFun");

createSDKLayerFun();

break;

}

},

initMap: function () {

this.upInitMonster();

},

createMonster: function (i, j, monsterType) {

if (monsterType <= 0) return;

var monster = new SpecialMonsterNode(monsterType, i, j);

if (monster != null) {

monster.x = this.tileMapPos[i][j].x;

monster.y = this.tileMapPos[i][j].y;

this.touchPanel.addChild(monster, Global.classicModeHeight - (i - 3));

monster.rowcol = [i, j];

monster.row = monster.rowcol[0];

this.tileMap.setGridInfo(i, j, monster);

}

},

fallMonstersCallFun: function () {

this.printArr();

this.searchConnectedMonsters(this.tileMap);

cc.log("//判断是否有相连（3个以上)");

var isConnected = this.checkIsConnectedMonsters();

if (isConnected) {

if (this.touchMonster == null) {

if (!this.isTouchDown) {

var isFallDowaning = this.searchIsFallDown(this.tileMap);

var isMonsterShake = this.searchIsShakeMonster(this.tileMap);

cc.log(this.tileMap.gridArr);

if (isFallDowaning && isMonsterShake) {

this.upInitMonster();

}

}

}

} else {

this.searchConnectedMonstersArrs();

this.shakeConnectedMonsters();

}

},

checkIsConnectedMonsters: function () {

if (this.tempMonsterArrs == null) return;

var connectedMonstersTimes = 0;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {

connectedMonstersTimes++;

}

}

}

}

if (connectedMonstersTimes == 0) {

return true;

} else {

return false;

}

},

createMonsterByRows: function (rows, upAndDown) {

var monsterTypeArr = [];

for (var j = 0; j < rows; j++) {

for (var i = 0; i < 7; i++) {

var monsterType = randomNums(1, 9);

if (monsterType == 7) {

monsterType = 6;

}

if (this.LV < 5) {

if (monsterType == 6) {

monsterType = randomNums(1, 6)

}

if (monsterType == 8) {

monsterType = randomNums(1, 6);

}

}

if (this.LV >= 5 && this.LV < 7) {

if (monsterType == 8) {

monsterType = randomNums(1, 6);

}

}

if (i >= 2) {

if (monsterTypeArr[i - 1] == monsterType && monsterTypeArr[i - 2] == monsterType) {

if (monsterType != 1) {

monsterType = monsterType - 1;

} else {

monsterType = 2;

}

}

}

monsterTypeArr.push(monsterType);

if (upAndDown == 1) {

if (this.tileMap.gridArr[Global.classicModeHeight - 1][i] == null && this.tileMap.gridArr[Global.classicModeHeight - 2][i] == null) {

var monster = new SpecialMonsterNode(monsterType, Global.classicModeHeight - 1, i);

monster.x = this.tileMapPos[Global.classicModeHeight - 1][i].x;

monster.y = this.tileMapPos[Global.classicModeHeight - 1][i].y;

monster.isCanFallDown = false;

this.touchPanel.addChild(monster);

monster.rowcol = [Global.classicModeHeight - 1, i];

this.tileMap.setGridInfo(Global.classicModeHeight - 1, i, monster);

}

}

if (upAndDown == 2) {

if (this.tileMap.gridArr[3 - 1 - j + 1][i] == null) {

var monster = new SpecialMonsterNode(monsterType, 3 - 1 - j + 1, i);

monster.x = this.tileMapPos[3 - 1 - j][i].x;

monster.y = this.tileMapPos[3 - 1 - j][i].y;

monster.isCanFallDown = true;

this.touchPanel.addChild(monster);

this.tileMap.setGridInfo(3 - 1 - j, i, monster);

this.gameControl.upMonster(3 - 1 - j, i, monster, rows);

}

}

}

}

},

monsterCallFun: function (target, data) {

if (data == null) return;

var monster = data[0];

var j = data[1];

var i = data[2];

monster.rowcol = [j, i];

monster.setLocalZOrder(Global.classicModeHeight - (j - 3));

monster.row = monster.rowcol[0];

cc.log("monster.rowcol :" + monster.rowcol);

this.tileMap.setGridInfo(j, i, monster);

},

searchRowsOfTileMap: function (tileMap, isFull) {

if (tileMap == null) return;

var tempNum = 0;

var num = 0;

if (isFull) {

for (var i = 3; i < tileMap.gridMaxY - 1; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

num++;

}

if (j == tileMap.gridMaxX - 1) {

if (num == tileMap.gridMaxX) {

tempNum++;

}

num = 0;

}

}

}

} else {

for (var \_i = 3; \_i < tileMap.gridMaxY - 1; \_i++) {

for (var \_j = 0; \_j < tileMap.gridMaxX; \_j++) {

if (tileMap.gridArr[\_i][\_j] != null) {

tempNum++;

break;

}

}

}

}

return tempNum;

},

searchColsOfTileMap: function (tileMap) {

if (tileMap == null) return;

var tempNum = [];

for (var i = 0; i < tileMap.gridMaxX; i++) {

if (tileMap.gridArr[tileMap.gridMaxY - 1 - 1][i] != null) {

tempNum.push(i);

}

}

return tempNum;

},

searchConnectedMonsters: function (tileMap) {

this.tempMonsterArrs = [];

var arr = [];

for (var i = 3; i < tileMap.gridMaxY - 2; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

this.tempMonsterArr = [];

if (tileMap.gridArr[i][j] != null) {

if (tileMap.gridArr[i][j].isMonster || tileMap.gridArr[i][j].isTimeMonster || tileMap.gridArr[i][j].isCageMonster || tileMap.gridArr[i][j].isBoxMonster || tileMap.gridArr[i][j].isAdvancedBoxMonster || tileMap.gridArr[i][j].isMonsterKing) {

if (!tileMap.gridArr[i][j].isSearched && !tileMap.gridArr[i][j].isWaiteDie && !tileMap.gridArr[i][j].isDown && !tileMap.gridArr[i][j].isUp) {

if (i == 3 || tileMap.gridArr[i - 1][j] != null) {

tileMap.gridArr[i][j].isSearched = true;

this.tempMonsterArr.push(tileMap.gridArr[i][j]);

this.searchMonstersByDir(i, j, tileMap, 0, 1);

this.searchMonstersByDir(i, j, tileMap, 0, -1);

this.searchMonstersByDir(i, j, tileMap, -1, 0);

this.searchMonstersByDir(i, j, tileMap, 1, 0);

this.tempMonsterArrs.push(this.tempMonsterArr);

}

}

}

}

}

}

for (var i = 3; i < tileMap.gridMaxY - 2; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

tileMap.gridArr[i][j].isSearched = false;

}

}

}

},

searchMonstersByDir: function (i, j, tileMap, horizontal, vertical) {

if (i + vertical > 2 && i + vertical < 13) {

if (tileMap.isExistGrid(j + horizontal, i + vertical)) {

if (tileMap.gridArr[i + vertical][j + horizontal] != null) {

if (tileMap.gridArr[i + vertical][j + horizontal].isMonster || tileMap.gridArr[i + vertical][j + horizontal].isTimeMonster || tileMap.gridArr[i + vertical][j + horizontal].isCageMonster || tileMap.gridArr[i + vertical][j + horizontal].isBoxMonster || tileMap.gridArr[i + vertical][j + horizontal].isAdvancedBoxMonster || tileMap.gridArr[i + vertical][j + horizontal].isMonsterKing) {

if (tileMap.gridArr[i + vertical][j + horizontal].skin == tileMap.gridArr[i][j].skin || tileMap.gridArr[i + vertical][j + horizontal].isTimeMonster) {

if (!tileMap.gridArr[i + vertical][j + horizontal].isSearched && !tileMap.gridArr[i + vertical][j + horizontal].isWaiteDie && !tileMap.gridArr[i + vertical][j + horizontal].isDown && !tileMap.gridArr[i + vertical][j + horizontal].isUp) {

if (i + vertical == 3 || tileMap.gridArr[i + vertical - 1][j + horizontal] != null) {

tileMap.gridArr[i + vertical][j + horizontal].isSearched = true;

this.tempMonsterArr.push(tileMap.gridArr[i + vertical][j + horizontal]);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 0, 1);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 0, -1);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, 1, 0);

this.searchMonstersByDir(i + vertical, j + horizontal, tileMap, -1, 0);

}

}

}

}

}

}

}

},

searchConnectedMonstersArrs: function () {

if (this.tempMonsterArrs == null) return;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {

this.connectMonsterArrs.push(this.tempMonsterArrs[i]);

}

}

}

}

},

shakeConnectedMonsters: function () {

if (this.connectMonsterArrs == null) return;

var connectMonsterArrsLength = this.connectMonsterArrs.length;

for (var i = 0; i < connectMonsterArrsLength; i++) {

var plength = this.connectMonsterArrs[i].length;

if (plength >= 3) {

var skin = this.connectMonsterArrs[i][0].skin;

cc.log("shake 相连的怪物");

var isExistArr = false;

var shakeTimes = Global.monsterShakeTimes;

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (this.connectMonsterArrs[i][j].isShake) {

var removeAct = this.gameControl.searchExistArrFromRemoveList(this.connectMonsterArrs[i][j]);

isExistArr = true;

break;

}

}

if (isExistArr) {

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (!this.connectMonsterArrs[i][j].isShake) {

if (removeAct != null) {

if (removeAct.arr != null) {

this.connectMonsterArrs[i][j].isShake = true;

playEffect(19);

this.connectMonsterArrs[i][j].playShakeAni();

removeAct.times = removeAct.times + 5;

removeAct.arr.push(this.connectMonsterArrs[i][j]);

playEffect(20);

}

}

}

}

} else {

var isMonsterRemove = false;

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (this.connectMonsterArrs[i][j].isRemve) {

this.connectMonsterArrs[i][j].isRemve = false;

isMonsterRemove = true;

}

}

for (var j = 0; j < plength; j++) {

if (this.connectMonsterArrs[i][j] == null) return;

if (!this.connectMonsterArrs[i][j].isShake) {

this.connectMonsterArrs[i][j].isShake = true;

this.connectMonsterArrs[i][j].playShakeAni();

if (j == plength - 1) {

playEffect(19);

}

}

}

this.gameControl.createRemoveList(this.connectMonsterArrs[i], shakeTimes, skin);

}

}

}

this.connectMonsterArrs = [];

},

removeConnectedMonsters: function () {

if (this.tempMonsterArrs == null) return;

this.tempColsArr = [0, 0, 0, 0, 0, 0, 0];

this.isClearMonster = true;

var tempMonsterArrLength = this.tempMonsterArrs.length;

if (this.tempMonsterArrs.length > 0) {

for (var i = 0; i < tempMonsterArrLength; i++) {

if (this.tempMonsterArrs[i] != null) {

var pLength = this.tempMonsterArrs[i].length;

if (pLength >= 3) {

cc.log("移出相连的怪物");

}

}

}

}

this.tempMonsterArr = [];

this.tempMonsterArrs = [];

},

fallMonstersBySpace: function () {

Global.monsterDown = false;

if (this.isBanFall) return;

this.setFallRemoveRows();

for (var i = 0; i < this.removeRows.length; i++) {

var row = this.removeRows[i];

var space = 0;

for (var j = 3; j < Global.classicModeHeight; j++) {

var monster = this.tileMap.gridArr[j][row];

if (monster == null || monster.isBox || monster.isAdvancedBox) {

space++;

} else {

if (space > 0) {

if (!monster.isShake && !monster.isDown && !monster.isWaiteDie && monster.isCanFallDown) {

if (monster.isMonster || monster.isIce || monster.isTimeMonster || monster.isCageMonster || monster.isMonsterKing) {

this.gameControl.moveMonster(j, row, monster, 1);

}

}

if (monster.isShake || monster.isBoxMonster || monster.isAdvancedBoxMonster || monster.isWaiteDie) {

space = 0;

}

}

}

}

}

this.removeRows = [];

if (!Global.monsterDown) {

if (this.gameControl.isMoveEnd) {

if (currentScene.gameLayer.isTouchOnMouseUp) {

currentScene.gameLayer.isTouchOnMouseUp = false;

}

cc.log("this.gameControl.isMoveEnd");

var self = this;

var callAction = cc.callFunc(function () {

self.fallMonstersCallFun();

});

this.runAction(cc.sequence(cc.delayTime(0.2), callAction));

} else {

if (currentScene.gameLayer.isTouchOnMouseUp) {

currentScene.gameLayer.isTouchOnMouseUp = false;

cc.log("currentScene.gameLayer.isTouchOnMouseUp +++++++++++++++++++++++ this.gameControl.isMoveEnd");

var self = this;

var callAction = cc.callFunc(function () {

self.fallMonstersCallFun();

});

this.runAction(cc.sequence(cc.delayTime(0.2), callAction));

}

}

}

},

fallMonster: function () {

cc.log("每隔一段时间下落 一排怪物");

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver) return;

var self = this;

var callFun = cc.callFunc(function () {

self.fallMonstersBySpace();

});

var callFunI = cc.callFunc(function () {

self.createMonsterByRows(1, 1);

});

var callFunII = cc.callFunc(function () {

self.setMonsterCanFallDown();

});

var callFunIII = cc.callFunc(function () {

self.playMonsterShake();

});

var callFunIIII = cc.callFunc(function () {

self.stopMonsterShake();

});

this.runAction(cc.sequence(callFunIII, cc.delayTime(0.5), callFunIIII, callFunII, callFun, cc.delayTime(0.5), callFunI));

},

setMonsterCanFallDown: function () {

for (var i = 0; i < 7; i++) {

var monster = this.tileMap.gridArr[Global.classicModeHeight - 1][i];

if (monster != null) {

monster.isCanFallDown = true;

}

}

},

playMonsterShake: function () {

for (var i = 0; i < 7; i++) {

var monster = this.tileMap.gridArr[Global.classicModeHeight - 1][i];

if (monster != null) {

var shake = monsterShakeAction(monster.rowcol);

var shakeAction = cc.sequence(shake[0], shake[1], shake[2], shake[3], shake[4], shake[5], shake[6], shake[7], shake[8]).repeatForever();

shakeAction.setTag(5000);

monster.runAction(shakeAction);

}

}

},

stopMonsterShake: function () {

for (var i = 0; i < 7; i++) {

var monster = this.tileMap.gridArr[Global.classicModeHeight - 1][i];

if (monster != null) {

monster.stopActionByTag(5000);

}

}

},

upInitMonster: function () {

if (currentScene.gameLayer.isTouchDown || currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver) return;

if (currentScene.gameControl.moveList.length != 0 || currentScene.gameControl.removeList.length != 0 || currentScene.gameControl.removeSpecialMonsterList.length != 0 || currentScene.gameControl.upMonsterList.length > 1 || currentScene.gameControl.downMonsterList.length != 0) {

if (currentScene.gameControl.removeSpecialMonsterList.length != 0 || currentScene.gameControl.removeList.length != 0) {

return;

} else {

return;

}

}

if (this.isCreateMonsterDown) return;

var MonsterRowsCols = this.computeCreateMonsterDownRows(this.tileMap);

var rows = MonsterRowsCols[0];

if (rows == 0) {

cc.log("upInitMonster");

this.printArr();

this.searchConnectedMonsters(this.tileMap);

var isConnected = this.checkIsConnectedMonsters();

if (!isConnected) {

this.searchConnectedMonstersArrs();

this.shakeConnectedMonsters();

}

return;

}

this.isCreateMonsterDown = true;

var cols = MonsterRowsCols[1];

var tileMapRows = this.searchRowsOfTileMap(this.tileMap, false);

for (var i = tileMapRows + 3; i >= 3; i--) {

for (var j = 0; j < Global.classicModeWidth; j++) {

if (i < Global.classicModeHeight - 1 && i + 1 <= Global.classicModeHeight - 2) {

if (this.tileMap.gridArr[i + 1][j] == null) {

var tileMapMonster = this.tileMap.gridArr[i][j];

if (tileMapMonster != null && !tileMapMonster.isBox && !tileMapMonster.isBoxMonster && !tileMapMonster.isAdvancedBox && !tileMapMonster.isAdvancedBoxMonster) {

if (!tileMapMonster.isDown && !tileMapMonster.isShake && !tileMapMonster.isWaiteDie && !tileMapMonster.isUp) {

this.gameControl.upMonster(i, j, tileMapMonster, rows);

}

}

}

}

}

}

this.createMonsterByRows(rows, 2);

},

getPosByGrid: function (Grid) {

return this.tileMapPos[Grid[1]][Grid[0]];

},

addTouchListener: function () {

engine.gameEvent.createMouseDown(this.touchPanel, this.onMouseDown);

engine.gameEvent.createMouseMove(this.touchPanel, this.onMouseMove);

engine.gameEvent.createMouseUp(this.touchPanel, this.onMouseUp);

},

onMouseDown: function (touch, event) {

currentScene.gameLayer.isTouchDown = true;

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver || currentScene.gameLayer.isBanFall) return;

cc.log("onTouchBegan");

var pos = touch.getLocation();

if (currentScene.gameLayer.touchMonster == null) {

currentScene.gameLayer.selectTouchMonster(pos);

}

return true;

},

selectTouchMonster: function (pos) {

currentScene.gameLayer.touchPosition = pos;

if (!currentScene.gameLayer.checkTouchArea(pos)) return;

cc.log("上升时，不能移动+++++++");

cc.log("上升时，不能移动-------");

var tilePos = currentScene.gameLayer.getTileMapPosByPoint(pos);

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, tilePos);

grid[1] = grid[1] + 3;

if ((currentScene.gameLayer.isSelectDyeBottle || currentScene.gameLayer.isSelectToiletSucker) && !currentScene.gameLayer.isDyeing && !currentScene.gameLayer.isPlayToilet) {

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] == null && currentScene.gameLayer.isSelectToiletSucker) {

currentScene.gameLayer.isPlayToilet = false;

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

} else {

var props = new PropsNode();

if (currentScene.gameLayer.isSelectDyeBottle) {

props.type = 1;

currentScene.gameLayer.isDyeing = true;

props.initProps(res.dyeingbottle\_json, "dyeingbottle", currentScene.gameLayer, grid);

}

if (currentScene.gameLayer.isSelectToiletSucker) {

props.type = 2;

currentScene.gameLayer.isPlayToilet = true;

props.initProps(res.toiletplug\_json, "toiletplug", currentScene.gameLayer, grid);

}

var tempPos = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]];

props.setPosition(tempPos);

props.x = props.x + 3;

props.y = props.y + 180;

currentScene.gameLayer.addChild(props, 50);

}

} else {

currentScene.gameLayer.frontGrid = grid;

if (!currentScene.gameLayer.checkMonsterMove(grid)) {

currentScene.gameLayer.isTouchDown = true;

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

var monster = currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]], 1, tilePos);

currentScene.gameLayer.touchMonster = monster;

} else if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

var monster = currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]], 2, tilePos);

currentScene.gameLayer.touchMonster = monster;

} else {

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = true;

currentScene.gameLayer.touchMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];

}

}

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

return;

}

if (currentScene.gameLayer.touchMonster.isMonster || currentScene.gameLayer.touchMonster.isTimeMonster || currentScene.gameLayer.touchMonster.isMonsterKing) {

if (currentScene.gameLayer.touchMonster.isFalling) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

return;

}

if (currentScene.gameLayer.touchMonster.isShake) currentScene.gameLayer.touchMonster = null;

if (currentScene.gameLayer.touchMonster != null) {

currentScene.gameLayer.touchMonster.isDown = false;

currentScene.gameLayer.touchMonster.isMove = true;

if (!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox && !currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox) {

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] = null;

}

if (currentScene.gameLayer.removeRows.indexOf(grid[0]) == -1) {

currentScene.gameLayer.removeRows.push(grid[0]);

}

}

}

}

}

},

onMouseMove: function (touch, event) {

if (currentScene.gameLayer.isGameWin || currentScene.gameLayer.isGameOver || currentScene.gameLayer.isBanFall) return;

cc.log("onTouchMoved");

var pos = touch.getLocation();

if (!currentScene.gameLayer.checkTouchArea(pos)) return;

if (currentScene.gameLayer.touchMonster == null) {

currentScene.gameLayer.selectTouchMonster(pos);

}

var delta = touch.getDelta();

var tilePos = currentScene.gameLayer.getTileMapPosByPoint(pos);

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) return;

var currentPoint = currentScene.gameLayer.touchMonster.getPosition();

var deltaX = tilePos.x - currentPoint.x;

var deltaY = tilePos.y - currentPoint.y;

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, tilePos);

grid[1] = grid[1] + 3;

currentScene.gameLayer.currentGrid = grid;

if (currentScene.gameLayer.touchPosition != null) {

if ((currentScene.gameLayer.currentGrid[0] != currentScene.gameLayer.frontGrid[0] || currentScene.gameLayer.currentGrid[1] != currentScene.gameLayer.frontGrid[1]) && (Math.abs(tilePos.x - currentScene.gameLayer.touchPosition.x) > Global.tileMapWidth)) {

currentScene.gameLayer.touchPosition = null;

currentScene.gameLayer.fallMonstersBySpace();

}

}

var monster = null;

if (grid[0] >= 0 && grid[0] < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 1) {

monster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];

}

var dirMonster = null;

if (currentScene.gameLayer.checkOutScreen(tilePos)) {

if (currentScene.gameLayer.checkMonsterMove(grid)) return;

if (monster != null) {

currentScene.gameLayer.isTouchInBox = true;

var tilePosDeltaX = tilePos.x - currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;

if (Math.abs(tilePosDeltaX) <= 15) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 1 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (dirMonster != null) {

cc.log("上左右 都已满");

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

}

if (tilePosDeltaX < 0 && Math.abs(tilePosDeltaX) > 15 && Math.abs(tilePosDeltaX) < 45) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 1 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (dirMonster != null) {

cc.log("上左右 都已满");

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

}

if (tilePosDeltaX > 0 && Math.abs(tilePosDeltaX) > 15 && Math.abs(tilePosDeltaX) < 45) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1];

if (dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]];

if (grid[1] + 1 >= Global.classicModeHeight - 1 || dirMonster != null) {

dirMonster = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1];

if (grid[0] - 1 < 0 || dirMonster != null) {

cc.log("上左右 都已满");

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]].y;

}

}

} else {

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1].y;

}

}

}

currentScene.gameLayer.touchMonster.setPosition(tilePos);

} else {

if (currentScene.gameLayer.isTouchInBox) {

currentScene.gameLayer.isTouchInBox = false;

if (currentScene.gameLayer.tileMapPos[grid[1]][grid[0]] != null) {

tilePos.x = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;

tilePos.y = currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].y;

}

} else {

if (deltaX > 0) {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 1, 0)) {

tilePos.x = currentPoint.x;

}

} else {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, -1, 0)) {

tilePos.x = currentPoint.x;

}

}

if (deltaY > 0) {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 0, 1)) {

tilePos.y = currentPoint.y;

}

} else {

if (currentScene.gameLayer.checkMonsterByDirPos(currentPoint, 0, -1)) {

tilePos.y = currentPoint.y;

}

}

}

currentScene.gameLayer.touchMonster.setPosition(tilePos);

}

}

},

onMouseUp: function (touch, event) {

cc.log("onTouchEnded");

if (currentScene.gameLayer.touchMonster == null) return;

if (currentScene.gameLayer.touchMonster.isIce || currentScene.gameLayer.touchMonster.isCage || currentScene.gameLayer.touchMonster.isBox || currentScene.gameLayer.touchMonster.isAdvancedBox || currentScene.gameLayer.touchMonster.isCageMonster) {

currentScene.gameLayer.touchMonster = null;

currentScene.gameLayer.currentGrid = null;

currentScene.gameLayer.frontGrid = null;

return;

}

var currentPoint = currentScene.gameLayer.touchMonster.getPosition();

var grid = currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap, currentPoint);

grid[1] = grid[1] + 3;

var gridPos = currentScene.gameLayer.getPosByGrid(grid);

if (currentScene.gameLayer.touchMonster != null) {

currentScene.gameLayer.isTouchDown = false;

currentScene.gameLayer.touchMonster.isMove = false;

currentScene.gameLayer.touchMonster.setPosition(gridPos);

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] == null) {

cc.log("onMouseUp——该格子为空");

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

cc.log("onMouseUp——该格子不为空");

if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox) {

var skin = currentScene.gameLayer.touchMonster.skin;

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin + 1);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin = skin;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster = true;

} else if (currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]] != null && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox) {

var skin = currentScene.gameLayer.touchMonster.skin;

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin + 1);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin = skin;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol = [grid[1], grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster = true;

} else {

cc.log("该格子 已被占用");

if (grid[0] - 1 >= 0 && grid[0] - 1 < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 1 && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0] - 1]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].rowcol = [grid[1], grid[0] - 1];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] - 1].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

if (grid[0] + 1 >= 0 && grid[0] + 1 < Global.classicModeWidth && grid[1] >= 3 && grid[1] < Global.classicModeHeight - 1 && currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0] + 1]);

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].rowcol = [grid[1], grid[0] + 1];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].row = currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0] + 1].setLocalZOrder(Global.classicModeHeight - (grid[1] - 3));

} else {

if (grid[0] >= 0 && grid[0] < Global.classicModeWidth && grid[1] + 1 >= 3 && grid[1] + 1 < Global.classicModeHeight - 1 && currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]] == null) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1] + 1][grid[0]]);

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].rowcol = [grid[1] + 1, grid[0]];

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].row = currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[grid[1] + 1][grid[0]].setLocalZOrder(Global.classicModeHeight - (grid[1] + 1 - 3));

} else {

var isHaveEmptyGrid = false;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

if (currentScene.gameLayer.tileMap.gridArr[i][j] == null && !isHaveEmptyGrid) {

currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[i][j]);

currentScene.gameLayer.tileMap.gridArr[i][j] = currentScene.gameLayer.touchMonster;

currentScene.gameLayer.tileMap.gridArr[i][j].isDelete = false;

currentScene.gameLayer.tileMap.gridArr[i][j].rowcol = [i, j];

currentScene.gameLayer.tileMap.gridArr[i][j].row = currentScene.gameLayer.tileMap.gridArr[i][j].rowcol[0];

currentScene.gameLayer.tileMap.gridArr[i][j].setLocalZOrder(Global.classicModeHeight - (i + 1 - 3));

isHaveEmptyGrid = true;

}

}

}

if (!isHaveEmptyGrid) {

currentScene.gameLayer.touchMonster.removeFromParent();

currentScene.gameLayer.touchMonster = null;

}

}

}

}

}

}

}

currentScene.gameLayer.isTouchOnMouseUp = true;

currentScene.gameLayer.fallMonstersBySpace();

currentScene.gameLayer.touchMonster = null;

currentScene.gameLayer.currentGrid = null;

currentScene.gameLayer.frontGrid = null;

currentScene.gameLayer.isTouchDown = false;

},

setFallRemoveRows: function () {

this.removeRows = [];

for (var i = 0; i < Global.classicModeWidth; i++) {

this.removeRows.push(i);

}

},

checkTouchArea: function (pos) {

if (pos.x < 718 && pos.x > 2 && pos.y < 960 && pos.y > 180) {

return true;

} else {

return false;

}

},

getTileMapGirdByPoint: function (tileMap, pos) {

var posX = pos.x - 43;

var posY = pos.y;

var lx = parseInt(posX / tileMap.gridWidth);

var ly = parseInt(posY / tileMap.gridHeight);

return [lx, ly];

},

getTileMapPosByPoint: function (pos) {

return cc.p(pos.x - this.touchPanel.x, pos.y - this.touchPanel.y);

},

checkOutScreen: function (pos) {

if (pos.x - Global.monsterWidth / 2 > 25 && pos.y - Global.monsterHeight / 2 > 5 && pos.x + Global.monsterWidth / 2 < Global.classicModeWidth \* Global.tileMapWidth + 50 && pos.y + Global.monsterHeight / 2 < Global.classicModeHeight \* Global.tileMapHeight + 5) {

return true;

} else {

return false;

}

},

checkMonsterByDirPos: function (pos, horizontal, vertical) {

var lx = parseInt((pos.x + horizontal - 43 + horizontal \* this.tileMap.gridWidth / 2) / this.tileMap.gridWidth);

var ly = parseInt((pos.y + vertical + vertical \* this.tileMap.gridHeight / 2) / this.tileMap.gridHeight);

ly = ly + 3;

if (this.tileMap.gridArr[ly][lx] == null) {

return false;

} else {

return true;

}

},

checkMonsterMove: function (gird) {

var bool1 = this.checkGrid(gird[0], gird[1] - 1);

var bool2 = this.checkGrid(gird[0] - 1, gird[1]);

var bool3 = this.checkGrid(gird[0] + 1, gird[1]);

var bool4 = this.checkGrid(gird[0], gird[1] + 1);

var bool = bool1 + bool2 + bool3 + bool4;

if (bool == 4) {

return true;

} else {

return false;

}

},

checkGrid: function (gird0, gird1) {

if (gird0 >= 0 && gird0 < Global.classicModeWidth && gird1 >= 3 && gird1 < Global.classicModeHeight - 1) {

if (this.tileMap.gridArr[gird1][gird0] != null) {

return 1;

} else {

return 0;

}

} else {

return 1;

}

},

computeCreateMonsterDownRows: function (tileMap) {

if (tileMap == null) return;

var allMonsterNum = 0;

var rows = 0;

for (var i = 3; i < tileMap.gridMaxY - 1; i++) {

for (var j = 0; j < tileMap.gridMaxX; j++) {

if (tileMap.gridArr[i][j] != null) {

allMonsterNum++;

}

}

}

if (allMonsterNum > 32) {

rows = 0;

} else {

if (allMonsterNum >= 21 && allMonsterNum < 28) {

rows = 0;

}

if (allMonsterNum >= 14 && allMonsterNum < 21) {

rows = 1;

}

if (allMonsterNum >= 7 && allMonsterNum < 14) {

rows = 1;

}

if (allMonsterNum < 7) {

rows = 1;

}

}

var cols = this.searchColsOfTileMap(tileMap);

return [rows, cols];

},

searchIsFallDown: function (tileMap) {

var times = 0;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = tileMap.gridArr[i][j];

if (monster != null) {

if (monster.isFalling) {

times++;

}

}

}

}

if (times == 0) {

return true;

} else {

return false;

}

},

searchIsShakeMonster: function (tileMap) {

var times = 0;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = tileMap.gridArr[i][j];

if (monster != null) {

if (monster.isShake) {

times++;

}

}

}

}

if (times == 0) {

return true;

} else {

return false;

}

},

printArr: function () {

for (var i = Global.classicModeHeight - 1; i > 2; i--) {

var temp = "";

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null) {

switch (monster.skin) {

case 0:

temp = temp + "" + "蓝";

break;

case 1:

temp = temp + "" + "绿";

break;

case 2:

temp = temp + "" + "紫";

break;

case 3:

temp = temp + "" + "红";

break;

case 4:

temp = temp + "" + "黄";

break;

}

} else {

temp = temp + "" + 0;

}

}

cc.log(temp);

}

},

starlightMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

for (var i = -1; i < 2; i++) {

for (var j = -1; j < 2; j++) {

var tempMonster = this.tileMap.gridArr[colRow[0] + i][colRow[1] + j];

if (colRow[0] + i >= 3 && colRow[0] + i < Global.classicModeHeight - 1 && colRow[1] + j >= 0 && colRow[1] + j < Global.classicModeWidth && tempMonster != null) {

if (i == 0 && j == 0) {

cc.log("自己已经被删除");

} else {

if (!tempMonster.isShake && !tempMonster.isWaiteDie && !tempMonster.isDown && !tempMonster.isUp) {

playEffect(22);

tempMonster.initDieEffectsType(1);

if (tempMonster.effectsType == 1) {

tempMonster.flameMonsterDie();

} else {

tempMonster.playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(tempMonster);

}

}

}

}

}

},

flameMonsterFall: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

if (monster.row == 3) {

cc.log("移到底炸周边六个");

monster.playDieAni();

} else {

if (this.tileMap.gridArr[monster.row - 1][colRow[1]] != null && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isShake && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isWaiteDie && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isDown && !this.tileMap.gridArr[monster.row - 1][colRow[1]].isUp) {

playEffect(22);

this.tileMap.gridArr[monster.row - 1][colRow[1]].initDieEffectsType(6);

if (this.tileMap.gridArr[monster.row - 1][colRow[1]].effectsType == 1) {

this.tileMap.gridArr[monster.row - 1][colRow[1]].flameMonsterDie();

} else {

this.tileMap.gridArr[monster.row - 1][colRow[1]].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[monster.row - 1][colRow[1]]);

}

var moveAction = cc.moveBy(0.03, cc.p(0, -90));

var self = this;

var callBackAction = cc.callFunc(function () {

monster.row = monster.row - 1;

self.flameMonsterFall(monster);

});

monster.runAction(cc.sequence(cc.delayTime(0.05), moveAction, callBackAction));

}

},

flameMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

var row = monster.row;

if (row == 3) {

cc.log("移到底炸周边六个");

if (colRow[1] - 1 >= 0 && colRow[1] - 1 < Global.classicModeWidth && this.tileMap.gridArr[row][colRow[1] - 1] != null) {

if (!this.tileMap.gridArr[row][colRow[1] - 1].isWaiteDie && !this.tileMap.gridArr[row][colRow[1] - 1].isShake && !this.tileMap.gridArr[row][colRow[1] - 1].isDown && !this.tileMap.gridArr[row][colRow[1] - 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row][colRow[1] - 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row][colRow[1] - 1].effectsType == 1) {

this.tileMap.gridArr[row][colRow[1] - 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row][colRow[1] - 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1] - 1]);

}

}

if (colRow[1] - 1 >= 0 && colRow[1] - 1 < Global.classicModeWidth && row + 1 >= 3 && row + 1 < Global.classicModeHeight - 1 && this.tileMap.gridArr[row + 1][colRow[1] - 1] != null) {

if (!this.tileMap.gridArr[row + 1][colRow[1] - 1].isWaiteDie && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isShake && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isDown && !this.tileMap.gridArr[row + 1][colRow[1] - 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row + 1][colRow[1] - 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row + 1][colRow[1] - 1].effectsType == 1) {

this.tileMap.gridArr[row + 1][colRow[1] - 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row + 1][colRow[1] - 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row + 1][colRow[1] - 1]);

}

}

if (colRow[1] + 1 >= 0 && colRow[1] + 1 < Global.classicModeWidth && this.tileMap.gridArr[row][colRow[1] + 1] != null) {

if (!this.tileMap.gridArr[row][colRow[1] + 1].isWaiteDie && !this.tileMap.gridArr[row][colRow[1] + 1].isShake && !this.tileMap.gridArr[row][colRow[1] + 1].isDown && !this.tileMap.gridArr[row][colRow[1] + 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row][colRow[1] + 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row][colRow[1] + 1].effectsType == 1) {

this.tileMap.gridArr[row][colRow[1] + 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row][colRow[1] + 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1] + 1]);

}

}

if (colRow[1] + 1 >= 0 && colRow[1] + 1 < Global.classicModeWidth && row + 1 >= 3 && row + 1 < Global.classicModeHeight - 1 && this.tileMap.gridArr[row + 1][colRow[1] + 1] != null) {

if (!this.tileMap.gridArr[row + 1][colRow[1] + 1].isWaiteDie && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isShake && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isDown && !this.tileMap.gridArr[row + 1][colRow[1] + 1].isUp) {

playEffect(22);

this.tileMap.gridArr[row + 1][colRow[1] + 1].initDieEffectsType(6);

if (this.tileMap.gridArr[row + 1][colRow[1] + 1].effectsType == 1) {

this.tileMap.gridArr[row + 1][colRow[1] + 1].flameMonsterDie();

} else {

this.tileMap.gridArr[row + 1][colRow[1] + 1].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row + 1][colRow[1] + 1]);

}

}

}

},

lightningMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

for (var i = 0; i < Global.classicModeWidth; i++) {

if (i != colRow[1] && this.tileMap.gridArr[colRow[0]][i] != null) {

if (!this.tileMap.gridArr[colRow[0]][i].isWaiteDie && !this.tileMap.gridArr[colRow[0]][i].isShake && !this.tileMap.gridArr[colRow[0]][i].isDown && !this.tileMap.gridArr[colRow[0]][i].isUp) {

playEffect(22);

this.tileMap.gridArr[colRow[0]][i].initDieEffectsType(5);

if (this.tileMap.gridArr[colRow[0]][i].effectsType == 1) {

this.tileMap.gridArr[colRow[0]][i].flameMonsterDie();

} else {

this.tileMap.gridArr[colRow[0]][i].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[colRow[0]][i]);

}

}

}

for (var j = 0; j < 9; j++) {

if (j + 3 != colRow[0] && this.tileMap.gridArr[j + 3][colRow[1]] != null) {

if (!this.tileMap.gridArr[j + 3][colRow[1]].isWaiteDie && !this.tileMap.gridArr[j + 3][colRow[1]].isShake && !this.tileMap.gridArr[j + 3][colRow[1]].isDown && !this.tileMap.gridArr[j + 3][colRow[1]].isUp) {

playEffect(22);

this.tileMap.gridArr[j + 3][colRow[1]].initDieEffectsType(5);

if (this.tileMap.gridArr[j + 3][colRow[1]].effectsType == 1) {

this.tileMap.gridArr[j + 3][colRow[1]].flameMonsterDie();

} else {

this.tileMap.gridArr[j + 3][colRow[1]].playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[j + 3][colRow[1]]);

}

}

}

},

blackHoleMonsterBoom: function (monster) {

if (monster == null) return;

var colRow = monster.rowcol;

var self = this;

for (var j = colRow[0] - 2; j < colRow[0] + 3; j++) {

for (var i = colRow[1] - 2; i < colRow[1] + 3; i++) {

if (j >= 3 && j < Global.classicModeHeight - 2 && i >= 0 && i < Global.classicModeWidth && !(i == colRow[0] && j == colRow[1])) {

var rotaMonster = this.tileMap.gridArr[j][i];

if (rotaMonster != null && !rotaMonster.isWaiteDie && !rotaMonster.isShake && !rotaMonster.isDown && !rotaMonster.isUp) {

cc.log("//播放吸入动画");

rotaMonster.isWaiteDie = true;

var pStart = rotaMonster.getPosition();

var pEnd = monster.getPosition();

rotaMonster.setMonsterAngle(pStart, pEnd);

rotaMonster.addScheduleTime();

rotaMonster.setMonsterData(pStart, pEnd);

}

}

}

}

},

useDyeingBottle: function (grid) {

var skin = randomNums(0, 5);

for (var i = -1; i < 2; i++) {

for (var j = -1; j < 2; j++) {

if (grid[0] + j >= 0 && grid[0] + j < Global.classicModeWidth && grid[1] + i >= 3 && grid[1] + i < Global.classicModeHeight - 1 && this.tileMap.gridArr[grid[1] + i][grid[0] + j] != null) {

var monster = this.tileMap.gridArr[grid[1] + i][grid[0] + j];

if (!monster.isShake && (monster.isMonster || monster.isCageMonster)) {

monster.monster.setSkin(Global.monsterSkin[skin]);

monster.skin = skin;

}

}

}

}

gamePlayerData.money = gamePlayerData.money - 500;

currentScene.gameLayer.refreshCurrentMoney();

currentScene.gameLayer.isDyeing = false;

currentScene.gameLayer.isSelectDyeBottle = false;

currentScene.gameLayer.rootNode.dyebottlex\_1.visible = false;

this.fallMonstersCallFun();

},

useToiletPlug: function (grid) {

cc.log("useToiletPlug");

if (this.tileMap.gridArr[grid[1]][grid[0]] != null) {

var skin = this.tileMap.gridArr[grid[1]][grid[0]].skin;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null) {

if (monster.skin != null && monster.skin == skin) {

monster.initDieEffectsType(1);

if (monster.effectsType == 1) {

monster.flameMonsterDie();

} else {

monster.playDieAni();

}

this.gameControl.createRemoveSpecialMonsterList(monster);

}

}

}

}

}

gamePlayerData.money = gamePlayerData.money - 200;

currentScene.gameLayer.refreshCurrentMoney();

currentScene.gameLayer.isPlayToilet = false;

currentScene.gameLayer.isSelectToiletSucker = false;

currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible = false;

},

removedFromBox: function (monster, type, pos) {

monster.monster.removeFromParent();

monster.monster = null;

var ID = monster.skin + 1;

monster.skin = -1;

if (type == 1) {

monster.isBoxMonster = false;

monster.isBox = true;

}

if (type == 2) {

monster.isAdvancedBoxMonster = false;

monster.isAdvancedBox = true;

}

var tempMonster = new SpecialMonsterNode(ID);

if (tempMonster != null) {

tempMonster.x = pos.x;

tempMonster.y = pos.y;

this.touchPanel.addChild(tempMonster);

}

return tempMonster;

},

updateCheckout: function (dt) {

this.checkShowGameAlarm();

if (!this.isGameOver && this.checkIsFullMonster()) {

this.checkGameOver();

}

},

checkIsFullMonster: function () {

var monsterCount = 0;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null && !monster.isShake) {

monsterCount++;

}

}

}

if (monsterCount == 63) {

return true;

} else {

return false;

}

},

refreshTime: function (second, minute) {

var str = minute + ":" + second;

this.rootNode.BitmapFontLabel\_6.setString(str);

},

checkGameOver: function () {

var data = new Object();

data.lv = this.LV;

var time = new Date();

data.date = time.getFullYear() + "." + time.getMonth() + "." + time.getDate();

data.score = this.score;

gamePlayerData.gameRankingListData.push(data);

sortGameRankingListData();

if (this.score >= gamePlayerData.bestScore) {

gamePlayerData.bestScore = this.score;

}

Global.gameOverData = [];

Global.gameOverData.push(this.LV);

Global.gameOverData.push(this.score);

Global.gameOverData.push(gamePlayerData.bestScore);

cc.log("弹游戏结束界面");

this.isGameOver = true;

var callFunAction = cc.callFunc(function () {

currentScene.gameOver();

});

this.runAction(cc.sequence(cc.delayTime(1), callFunAction));

savePlayerData();

},

calculateGetStar: function () {

if (this.countTime >= this.gameStarThree) {

return 3;

} else if (this.countTime >= this.gameStarTwo && this.countTime < this.gameStarThree) {

return 2;

} else if (this.countTime >= this.gameStarOne && this.countTime < this.gameStarTwo) {

return 1;

} else {

cc.log("没有得星");

return 0;

}

},

showGameAlarm: function (boolean) {

if (boolean) {

this.rootNode.GameAlarm\_1.visible = true;

this.rootNode.GameAlarm\_1.runAction(cc.sequence(cc.fadeIn(0.2), cc.fadeOut(0.5)).repeatForever());

} else {

this.rootNode.GameAlarm\_1.visible = false;

}

},

checkShowGameAlarm: function () {

var monsterCount = 0;

for (var i = 3; i < Global.classicModeHeight - 1; i++) {

for (var j = 0; j < Global.classicModeWidth; j++) {

var monster = this.tileMap.gridArr[i][j];

if (monster != null && !monster.isShake) {

monsterCount++;

}

}

}

if (monsterCount > 45) {

if (!this.isShowGameAlarm) {

this.isShowGameAlarm = true;

this.showGameAlarm(this.isShowGameAlarm);

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 3);

}

} else {

if (this.isShowGameAlarm) {

this.unschedule(this.fallMonster);

this.schedule(this.fallMonster, 10);

}

this.isShowGameAlarm = false;

this.showGameAlarm(this.isShowGameAlarm);

}

},

showGameLevelUpNode: function () {

this.gameLevelUpNode.BitmapFontLabel\_1.setString("" + this.LV);

this.gameLevelUpNode.runAction(cc.sequence(cc.moveTo(1, cc.p(360, 640)), cc.delayTime(0.8), cc.moveTo(0.5, cc.p(360, 1920))));

}

});

var WinLayer = cc.Layer.extend({

rootNode: null,

home: null,

homePic: null,

replay: null,

replayPic: null,

continues: null,

continuesPic: null,

rainPanel: null,

gamewin: null,

getScore: 0,

getStar: 0,

getGold: 0,

ctor: function (score, star) {

this.\_super();

this.init(score, star);

},

init: function (score, star) {

this.initUI();

var self = this;

var callFunAction = cc.callFunc(function () {

self.initText(score, star);

self.initListener();

});

this.runAction(cc.sequence(cc.delayTime(1), callFunAction));

},

initUI: function () {

this.rootNode = loadCSS(res.GameWinLayer\_json);

this.addChild(this.rootNode);

this.rootNode.BitmapFontLabel\_1.setString("");

this.rootNode.BitmapFontLabel\_2.setString("");

this.rootNode.BitmapFontLabel\_1.visible = false;

this.rootNode.BitmapFontLabel\_2.visible = false;

var animationPanel = this.rootNode.Panel\_2;

cc.spriteFrameCache.addSpriteFrames(res.shengli\_plist);

cc.spriteFrameCache.addSpriteFrames(res.shengli1\_plist);

multipleAnimation(res.shengli\_json, res.shengli\_json);

this.gamewin = new GameSkeletalAnimation();

var data = skeletalAnimation[res.shengli\_json + "|shengli"];

data.playCount = 1;

this.gamewin.initialize(data);

this.gamewin.setFrameIntervalTime(24);

this.gamewin.isRemoveFromComplete = false;

animationPanel.addChild(this.gamewin);

this.gamewin.x = 380;

this.gamewin.y = 720;

this.home = new GameSkeletalAnimation();

var dataHome = skeletalAnimation[res.shengli\_json + "|gamehome"];

dataHome.playCount = 1;

this.home.initialize(dataHome);

this.home.setFrameIntervalTime(24);

this.home.isRemoveFromComplete = false;

animationPanel.addChild(this.home);

this.home.x = 200;

this.home.y = 510;

this.replay = new GameSkeletalAnimation();

var dataReplay = skeletalAnimation[res.shengli\_json + "|gamereplay"];

dataReplay.playCount = 1;

this.replay.initialize(dataReplay);

this.replay.setFrameIntervalTime(24);

this.replay.isRemoveFromComplete = false;

animationPanel.addChild(this.replay);

this.replay.x = 340;

this.replay.y = 450;

this.continues = new GameSkeletalAnimation();

var dataContinues = skeletalAnimation[res.shengli\_json + "|gamecontinues"];

dataContinues.playCount = 1;

this.continues.initialize(dataContinues);

this.continues.setFrameIntervalTime(24);

this.continues.isRemoveFromComplete = false;

animationPanel.addChild(this.continues);

this.continues.x = 500;

this.continues.y = 510;

this.homePic = this.home.getBimapByUrl("#shenglipng17.png");

this.replayPic = this.replay.getBimapByUrl("#shenglipng16.png");

this.continuesPic = this.continues.getBimapByUrl("#shenglipng15.png");

playEffect(6);

},

initText: function (score, star) {

this.getScore = score;

this.getStar = star;

this.getGold = star \* 5;

this.rootNode.BitmapFontLabel\_1.setString(score + "");

this.rootNode.BitmapFontLabel\_2.setString(this.getGold + "");

this.rootNode.BitmapFontLabel\_1.visible = true;

this.rootNode.BitmapFontLabel\_2.visible = true;

gamePlayerData.money += this.getGold;

currentScene.gameLayer.refreshCurrentMoney();

savePlayerData();

},

initListener: function () {

engine.gameEvent.createMouseDown(this.homePic, this.btnHomeFun);

engine.gameEvent.createMouseDown(this.replayPic, this.btnReplayFun);

engine.gameEvent.createMouseDown(this.continuesPic, this.btnContinuesFun);

},

showElement: function () {},

btnHomeFun: function () {

playEffect(2);

cc.log("btnHomeFun");

createCCSButtonScaling(this);

transitions(SelectScene);

},

btnReplayFun: function () {

playEffect(2);

cc.log("btnReplayFun");

createCCSButtonScaling(this);

if (Global.currentSelectLevel == 100) {

cc.log("Global.currentSelectLevel == 100");

} else {

Global.currentSelectLevel = Global.currentSelectLevel - 1;

}

transitions(GamePlayScene);

},

btnContinuesFun: function () {

playEffect(2);

cc.log("btnContinuesFun");

createCCSButtonScaling(this);

if (gamePlayerData.isJiHuo == 0) {

if (Global.currentSelectLevel == Global.jieshuoPage) {

createSDKLayerFun();

} else {

transitions(GamePlayScene);

}

} else {

transitions(GamePlayScene);

}

savePlayerData();

}

});

var DailyTaskLayer = cc.Layer.extend({

rootNode: null,

btnClose: null,

eliminate1: null,

button1: null,

eliminate2: null,

button2: null,

eliminate3: null,

button3: null,

onEnter: function () {

this.\_super();

this.init();

},

init: function () {

this.initUI();

this.initText();

this.initListener();

},

initText: function () {},

initUI: function () {

this.rootNode = loadCSS(res.dailyTaskLayer\_json);

this.addChild(this.rootNode);

this.btnClose = this.rootNode.Button\_1;

this.eliminate1 = this.rootNode.Eliminate;

this.eliminate1.Sprite\_6.initWithFile(Global.dailyTaskMonster[gamePlayerData.gameRemoveMonster[0]]);

this.eliminate1.BitmapFontLabel\_1.setString(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[0]] + "/" + gamePlayerData.gameRemoveMonster[1]);

this.button1 = this.eliminate1.Button\_2;

this.button1.tag = 1;

this.button1.setPressedActionEnabled(true);

this.eliminate2 = this.rootNode.Eliminate\_0;

this.eliminate2.Sprite\_6.initWithFile(Global.dailyTaskMonster[gamePlayerData.gameRemoveMonster[2]]);

this.eliminate2.BitmapFontLabel\_1\_0.setString(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[2]] + "/" + gamePlayerData.gameRemoveMonster[3]);

this.eliminate2.Image\_10.visible = false;

this.button2 = this.eliminate2.Button\_3;

this.button2.tag = 2;

this.button2.setPressedActionEnabled(true);

this.eliminate3 = this.rootNode.Eliminate\_1;

this.eliminate3.BitmapFontLabel\_1\_0.setString("" + gamePlayerData.gameRemoveMonster[4]);

this.eliminate3.Image\_10.visible = false;

this.button3 = this.eliminate3.Button\_6;

this.button3.tag = 3;

this.button3.setPressedActionEnabled(true);

if (gamePlayerData.isCompleteDailyTsk[0]) {

this.eliminate1.Image\_10.visible = true;

this.button1.visible = false;

} else {

this.eliminate1.Image\_10.visible = false;

this.button1.visible = true;

}

if (gamePlayerData.isCompleteDailyTsk[1]) {

this.eliminate2.Image\_10.visible = true;

this.button2.visible = false;

} else {

this.eliminate2.Image\_10.visible = false;

this.button2.visible = true;

}

if (gamePlayerData.isCompleteDailyTsk[2]) {

this.eliminate3.Image\_10.visible = true;

this.button3.visible = false;

} else {

this.eliminate3.Image\_10.visible = false;

this.button3.visible = true;

}

savePlayerData();

},

initListener: function () {

this.btnClose.addTouchEventListener(this.btnCloseFun);

this.button1.addTouchEventListener(this.btnTrueFun);

this.button2.addTouchEventListener(this.btnTrueFun);

this.button3.addTouchEventListener(this.btnTrueFun);

},

btnCloseFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnCloseFun");

if (currentScene.gameTaskLayer != null) {

currentScene.gameTaskLayer.removeFromParent();

currentScene.gameTaskLayer = null;

}

break;

}

},

btnTrueFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

cc.log("btnTrueFun" + sender.tag);

switch (sender.tag) {

case 1:

if (gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[0]] >= gamePlayerData.gameRemoveMonster[1]) {

playEffect(26);

gamePlayerData.money += 50;

currentScene.gameTaskLayer.button1.visible = false;

currentScene.gameTaskLayer.eliminate1.Image\_10.visible = true;

gamePlayerData.isCompleteDailyTsk[0] = true;

}

break;

case 2:

if (gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[2]] >= gamePlayerData.gameRemoveMonster[3]) {

playEffect(26);

gamePlayerData.money += 50;

currentScene.gameTaskLayer.button2.visible = false;

currentScene.gameTaskLayer.eliminate2.Image\_10.visible = true;

gamePlayerData.isCompleteDailyTsk[1] = true;

}

break;

case 3:

if (gamePlayerData.currentLevel >= gamePlayerData.gameRemoveMonster[4]) {

playEffect(26);

gamePlayerData.money += 50;

currentScene.gameTaskLayer.button3.visible = false;

currentScene.gameTaskLayer.eliminate3.Image\_10.visible = true;

gamePlayerData.isCompleteDailyTsk[2] = true;

}

break;

}

break;

var RankingListLayer = cc.Layer.extend({

rootNode: null,

btnBack: null,

scrollPanel: null,

onEnter: function () {

this.\_super();

this.init();

},

init: function () {

this.initUI();

this.initText();

this.initItem();

this.initListener();

},

initText: function () {},

initUI: function () {

this.rootNode = loadCSS(res.RankingListLayer\_json);

this.addChild(this.rootNode);

this.btnBack = this.rootNode.Button\_1;

this.scrollPanel = this.rootNode.Image\_1.ScrollView\_1;

},

initListener: function () {

this.btnBack.addTouchEventListener(this.btnBackFun);

},

initItem: function () {

for (var i = 0; i < 10; i++) {

var itemNode = ccs.load(res.RankingListNode\_json).node;

itemNode.x = 280;

itemNode.y = 1500 - i \* 160;

this.scrollPanel.addChild(itemNode);

if (i != 0) {

itemNode.getChildByName("Image\_2").getChildByName("Image\_3").visible = false;

}

var sp = cc.spriteFrameCache.getSpriteFrame("rank" + (i + 1) + ".png");

itemNode.getChildByName("Image\_2").getChildByName("Sprite\_2").setSpriteFrame(sp);

if (gamePlayerData.gameRankingListData[i] != null) {

itemNode.getChildByName("BitmapFontLabel\_1").setString("" + gamePlayerData.gameRankingListData[i].lv);

itemNode.getChildByName("BitmapFontLabel\_2").setString("" + gamePlayerData.gameRankingListData[i].date);

itemNode.getChildByName("BitmapFontLabel\_3").setString("" + gamePlayerData.gameRankingListData[i].score);

} else {

itemNode.getChildByName("BitmapFontLabel\_1").visible = false;

itemNode.getChildByName("BitmapFontLabel\_2").visible = false;

itemNode.getChildByName("BitmapFontLabel\_3").visible = false;

}

}

},

btnBackFun: function (sender, type) {

switch (type) {

case ccui.Widget.TOUCH\_ENDED:

playEffect(2);

cc.log("btnBackFun");

if (currentScene.gameRankLayer != null) {

currentScene.gameRankLayer.removeFromParent();

currentScene.gameRankLayer = null;

}

break;

}

}

});

}

savePlayerData();

}

});

ccui.HBox = ccui.Layout.extend({

ctor: function (size) {

ccui.Layout.prototype.ctor.call(this);

this.setLayoutType(ccui.Layout.LINEAR\_HORIZONTAL);

if (size) {

this.setContentSize(size);

}

}

});

ccui.HBox.create = function (size) {

return new ccui.HBox(size);

};

(function () {

if (cc === undefined) {

return;

}

var PluginManager = function () {};

PluginManager.prototype = {

constructor: PluginManager,

getInstance: function () {

return this;

},

loadPlugin: function (pluginName) {},

unloadPlugin: function (pluginName) {}

};

var PluginAssembly = function () {};

PluginAssembly.prototype = {

constructor: PluginAssembly,

setDebugMode: function (debug) {},

startSession: function (appKey) {},

setCaptureUncaughtException: function (Capture) {},

callFuncWithParam: function (funName) {

if (typeof this[funName] === 'function') {

return this[funName].apply(this, Array.prototype.splice.call(arguments, 1));

} else {

cc.log("function is not define");

}

},

callStringFuncWithParam: function (funName) {

this.callFuncWithParam.apply(arguments);

},

getPluginName: function () {

return this.\_name;

},

getPluginVersion: function () {

return this.\_version;

}

};

PluginAssembly.extend = function (name, porp) {

var p, prototype = {};

for (p in PluginAssembly.prototype) {

prototype[p] = PluginAssembly.prototype[p];

}

for (p in porp) {

prototype[p] = porp[p];

}

var tmp = eval("(function " + name + "Plugin(){})");

prototype.constructor = tmp;

tmp.prototype = prototype;

return tmp;

};

var Param = function (type, value) {

var paramType = plugin.PluginParam.ParamType,

tmpValue;

switch (type) {

case paramType.TypeInt:

tmpValue = parseInt(value);

break;

case paramType.TypeFloat:

tmpValue = parseFloat(value);

break;

case paramType.TypeBool:

tmpValue = Boolean(value);

break;

case paramType.TypeString:

tmpValue = String(value);

break;

case paramType.TypeStringMap:

tmpValue = value

break;

default:

tmpValue = value;

}

return tmpValue

};

Param.ParamType = {

TypeInt: 1,

TypeFloat: 2,

TypeBool: 3,

TypeString: 4,

TypeStringMap: 5

};

Param.AdsResultCode = {

AdsReceived: 0,

FullScreenViewShown: 1,

FullScreenViewDismissed: 2,

PointsSpendSucceed: 3,

PointsSpendFailed: 4,

NetworkError: 5,

UnknownError: 6

};

Param.PayResultCode = {

PaySuccess: 0,

PayFail: 1,

PayCancel: 2,

PayTimeOut: 3

};

Param.ShareResultCode = {

ShareSuccess: 0,

ShareFail: 1,

ShareCancel: 2,

ShareTimeOut: 3

};

var PluginList = {};

var Plugin = {

extend: function (name, extend) {

var config = (cc.game.config && cc.game.config.plugin) || {};

PluginList[name] = new(PluginAssembly.extend(name, extend));

typeof PluginList[name].ctor === "function"&& PluginList[name].ctor(config[name]);

},

PluginList: PluginList,

PluginParam: Param,

PluginManager: new PluginManager()

};

window.plugin = Plugin;

})();

ccui.Layout = ccui.Widget.extend({

\_clippingEnabled: false,

\_backGroundScale9Enabled: null,

\_backGroundImage: null,

\_backGroundImageFileName: null,

\_backGroundImageCapInsets: null,

\_colorType: null,

\_bgImageTexType: ccui.Widget.LOCAL\_TEXTURE,

\_colorRender: null,

\_gradientRender: null,

\_color: null,

\_startColor: null,

\_endColor: null,

\_alongVector: null,

\_opacity: 255,

\_backGroundImageTextureSize: null,

\_layoutType: null,

\_doLayoutDirty: true,

\_clippingRectDirty: true,

\_clippingType: null,

\_clippingStencil: null,

\_scissorRectDirty: false,

\_clippingRect: null,

\_clippingParent: null,

\_className: "Layout",

\_backGroundImageColor: null,

\_finalPositionX: 0,

\_finalPositionY: 0,

\_backGroundImageOpacity: 0,

\_loopFocus: false,

\_\_passFocusToChild: true,

\_isFocusPassing: false,

\_isInterceptTouch: false,

ctor: function () {

this.\_layoutType = ccui.Layout.ABSOLUTE;

this.\_widgetType = ccui.Widget.TYPE\_CONTAINER;

this.\_clippingType = ccui.Layout.CLIPPING\_SCISSOR;

this.\_colorType = ccui.Layout.BG\_COLOR\_NONE;

ccui.Widget.prototype.ctor.call(this);

this.ignoreContentAdaptWithSize(false);

this.setContentSize(cc.size(0, 0));

this.setAnchorPoint(0, 0);

this.onPassFocusToChild = this.\_findNearestChildWidgetIndex.bind(this);

this.\_backGroundImageCapInsets = cc.rect(0, 0, 0, 0);

this.\_color = cc.color(255, 255, 255, 255);

this.\_startColor = cc.color(255, 255, 255, 255);

this.\_endColor = cc.color(255, 255, 255, 255);

this.\_alongVector = cc.p(0, -1);

this.\_backGroundImageTextureSize = cc.size(0, 0);

this.\_clippingRect = cc.rect(0, 0, 0, 0);

this.\_backGroundImageColor = cc.color(255, 255, 255, 255);

},

onEnter: function () {

ccui.Widget.prototype.onEnter.call(this);

if (this.\_clippingStencil) this.\_clippingStencil.onEnter();

this.\_doLayoutDirty = true;

this.\_clippingRectDirty = true;

},

onExit: function () {

ccui.Widget.prototype.onExit.call(this);

if (this.\_clippingStencil) this.\_clippingStencil.onExit();

},

setLoopFocus: function (loop) {

this.\_loopFocus = loop;

},

isLoopFocus: function () {

return this.\_loopFocus;

},

setPassFocusToChild: function (pass) {

this.\_\_passFocusToChild = pass;

},

isPassFocusToChild: function () {

return this.\_\_passFocusToChild;

},

findNextFocusedWidget: function (direction, current) {

if (this.\_isFocusPassing || this.isFocused()) {

var parent = this.getParent();

this.\_isFocusPassing = false;

if (this.\_\_passFocusToChild) {

var w = this.\_passFocusToChild(direction, current);

if (w instanceof ccui.Layout && parent) {

parent.\_isFocusPassing = true;

return parent.findNextFocusedWidget(direction, this);

}

return w;

}

if (null == parent || !(parent instanceof ccui.Layout)) return this;

parent.\_isFocusPassing = true;

return parent.findNextFocusedWidget(direction, this);

} else if (current.isFocused() || current instanceof ccui.Layout) {

if (this.\_layoutType === ccui.Layout.LINEAR\_HORIZONTAL) {

switch (direction) {

case ccui.Widget.LEFT:

return this.\_getPreviousFocusedWidget(direction, current);

break;

case ccui.Widget.RIGHT:

return this.\_getNextFocusedWidget(direction, current);

break;

case ccui.Widget.DOWN:

case ccui.Widget.UP:

if (this.\_isLastWidgetInContainer(this, direction)) {

if (this.\_isWidgetAncestorSupportLoopFocus(current, direction)) return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

return current;

} else {

return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

}

break;

default:

cc.assert(0, "Invalid Focus Direction");

return current;

}

} else if (this.\_layoutType === ccui.Layout.LINEAR\_VERTICAL) {

switch (direction) {

case ccui.Widget.LEFT:

case ccui.Widget.RIGHT:

if (this.\_isLastWidgetInContainer(this, direction)) {

if (this.\_isWidgetAncestorSupportLoopFocus(current, direction)) return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

return current;

} else return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

break;

case ccui.Widget.DOWN:

return this.\_getNextFocusedWidget(direction, current);

break;

case ccui.Widget.UP:

return this.\_getPreviousFocusedWidget(direction, current);

break;

default:

cc.assert(0, "Invalid Focus Direction");

return current;

}

} else {

cc.assert(0, "Un Supported Layout type, please use VBox and HBox instead!!!");

return current;

}

} else return current;

},

onPassFocusToChild: null,

addChild: function (widget, zOrder, tag) {

if ((widget instanceof ccui.Widget)) {

this.\_supplyTheLayoutParameterLackToChild(widget);

}

ccui.Widget.prototype.addChild.call(this, widget, zOrder, tag);

this.\_doLayoutDirty = true;

},

removeChild: function (widget, cleanup) {

ccui.Widget.prototype.removeChild.call(this, widget, cleanup);

this.\_doLayoutDirty = true;

},

removeAllChildren: function (cleanup) {

ccui.Widget.prototype.removeAllChildren.call(this, cleanup);

this.\_doLayoutDirty = true;

},

removeAllChildrenWithCleanup: function (cleanup) {

ccui.Widget.prototype.removeAllChildrenWithCleanup.call(this, cleanup);

this.\_doLayoutDirty = true;

},

isClippingEnabled: function () {

return this.\_clippingEnabled;

},

visit: function (parentCmd) {

if (!this.\_visible) return;

this.\_adaptRenderers();

this.\_doLayout();

if (this.\_clippingEnabled) {

switch (this.\_clippingType) {

case ccui.Layout.CLIPPING\_STENCIL:

this.\_renderCmd.stencilClippingVisit(parentCmd);

break;

case ccui.Layout.CLIPPING\_SCISSOR:

this.\_renderCmd.scissorClippingVisit(parentCmd);

break;

default:

break;

}

} else {

ccui.Widget.prototype.visit.call(this, parentCmd);

}

},

setClippingEnabled: function (able) {

if (able === this.\_clippingEnabled) return;

this.\_clippingEnabled = able;

switch (this.\_clippingType) {

case ccui.Layout.CLIPPING\_SCISSOR:

case ccui.Layout.CLIPPING\_STENCIL:

if (able) {

this.\_clippingStencil = new cc.DrawNode();

this.\_renderCmd.rebindStencilRendering(this.\_clippingStencil);

if (this.\_running) this.\_clippingStencil.onEnter();

this.\_setStencilClippingSize(this.\_contentSize);

} else {

if (this.\_running && this.\_clippingStencil) this.\_clippingStencil.onExit();

this.\_clippingStencil = null;

}

break;

default:

break;

}

},

setClippingType: function (type) {

if (type === this.\_clippingType) return;

var clippingEnabled = this.isClippingEnabled();

this.setClippingEnabled(false);

this.\_clippingType = type;

this.setClippingEnabled(clippingEnabled);

},

getClippingType: function () {

return this.\_clippingType;

},

\_setStencilClippingSize: function (size) {

if (this.\_clippingEnabled) {

var rect = [];

rect[0] = cc.p(0, 0);

rect[1] = cc.p(size.width, 0);

rect[2] = cc.p(size.width, size.height);

rect[3] = cc.p(0, size.height);

var green = cc.color.GREEN;

this.\_clippingStencil.clear();

this.\_clippingStencil.setLocalBB && this.\_clippingStencil.setLocalBB(0, 0, size.width, size.height);

this.\_clippingStencil.drawPoly(rect, 4, green, 0, green);

}

},

\_getClippingRect: function () {

if (this.\_clippingRectDirty) {

var worldPos = this.convertToWorldSpace(cc.p(0, 0));

var t = this.getNodeToWorldTransform();

var scissorWidth = this.\_contentSize.width \* t.a;

var scissorHeight = this.\_contentSize.height \* t.d;

var parentClippingRect;

var parent = this;

while (parent) {

parent = parent.getParent();

if (parent && parent instanceof ccui.Layout && parent.isClippingEnabled()) {

this.\_clippingParent = parent;

break;

}

}

if (this.\_clippingParent) {

parentClippingRect = this.\_clippingParent.\_getClippingRect();

this.\_clippingRect.x = Math.max(worldPos.x, parentClippingRect.x);

this.\_clippingRect.y = Math.max(worldPos.y, parentClippingRect.y);

var right = Math.min(worldPos.x + scissorWidth, parentClippingRect.x + parentClippingRect.width);

var top = Math.min(worldPos.y + scissorHeight, parentClippingRect.y + parentClippingRect.height);

this.\_clippingRect.width = Math.max(0.0, right - this.\_clippingRect.x);

this.\_clippingRect.height = Math.max(0.0, top - this.\_clippingRect.y);

} else {

this.\_clippingRect.x = worldPos.x;

this.\_clippingRect.y = worldPos.y;

this.\_clippingRect.width = scissorWidth;

this.\_clippingRect.height = scissorHeight;

}

this.\_clippingRectDirty = false;

}

return this.\_clippingRect;

},

\_onSizeChanged: function () {

ccui.Widget.prototype.\_onSizeChanged.call(this);

var locContentSize = this.\_contentSize;

this.\_setStencilClippingSize(locContentSize);

this.\_doLayoutDirty = true;

this.\_clippingRectDirty = true;

if (this.\_backGroundImage) {

this.\_backGroundImage.setPosition(locContentSize.width \* 0.5, locContentSize.height \* 0.5);

if (this.\_backGroundScale9Enabled && this.\_backGroundImage instanceof ccui.Scale9Sprite) this.\_backGroundImage.setPreferredSize(locContentSize);

}

if (this.\_colorRender) this.\_colorRender.setContentSize(locContentSize);

if (this.\_gradientRender) this.\_gradientRender.setContentSize(locContentSize);

},

setBackGroundImageScale9Enabled: function (able) {

if (this.\_backGroundScale9Enabled === able) return;

this.removeProtectedChild(this.\_backGroundImage);

this.\_backGroundImage = null;

this.\_backGroundScale9Enabled = able;

this.\_addBackGroundImage();

this.setBackGroundImage(this.\_backGroundImageFileName, this.\_bgImageTexType);

this.setBackGroundImageCapInsets(this.\_backGroundImageCapInsets);

},

isBackGroundImageScale9Enabled: function () {

return this.\_backGroundScale9Enabled;

},

setBackGroundImage: function (fileName, texType) {

if (!fileName) return;

texType = texType || ccui.Widget.LOCAL\_TEXTURE;

if (this.\_backGroundImage === null) {

this.\_addBackGroundImage();

this.setBackGroundImageScale9Enabled(this.\_backGroundScale9Enabled);

}

this.\_backGroundImageFileName = fileName;

this.\_bgImageTexType = texType;

var locBackgroundImage = this.\_backGroundImage;

switch (this.\_bgImageTexType) {

case ccui.Widget.LOCAL\_TEXTURE:

locBackgroundImage.initWithFile(fileName);

break;

case ccui.Widget.PLIST\_TEXTURE:

locBackgroundImage.initWithSpriteFrameName(fileName);

break;

default:

break;

}

if (this.\_backGroundScale9Enabled) locBackgroundImage.setPreferredSize(this.\_contentSize);

this.\_backGroundImageTextureSize = locBackgroundImage.getContentSize();

locBackgroundImage.setPosition(this.\_contentSize.width \* 0.5, this.\_contentSize.height \* 0.5);

this.\_updateBackGroundImageColor();

},

setBackGroundImageCapInsets: function (capInsets) {

if (!capInsets) return;

var locInsets = this.\_backGroundImageCapInsets;

locInsets.x = capInsets.x;

locInsets.y = capInsets.y;

locInsets.width = capInsets.width;

locInsets.height = capInsets.height;

if (this.\_backGroundScale9Enabled) this.\_backGroundImage.setCapInsets(capInsets);

},

getBackGroundImageCapInsets: function () {

return cc.rect(this.\_backGroundImageCapInsets);

},

\_supplyTheLayoutParameterLackToChild: function (locChild) {

if (!locChild) {

return;

}

switch (this.\_layoutType) {

case ccui.Layout.ABSOLUTE:

break;

case ccui.Layout.LINEAR\_HORIZONTAL:

case ccui.Layout.LINEAR\_VERTICAL:

var layoutParameter = locChild.getLayoutParameter(ccui.LayoutParameter.LINEAR);

if (!layoutParameter) locChild.setLayoutParameter(new ccui.LinearLayoutParameter());

break;

case ccui.Layout.RELATIVE:

var layoutParameter = locChild.getLayoutParameter(ccui.LayoutParameter.RELATIVE);

if (!layoutParameter) locChild.setLayoutParameter(new ccui.RelativeLayoutParameter());

break;

default:

break;

}

},

\_addBackGroundImage: function () {

var contentSize = this.\_contentSize;

if (this.\_backGroundScale9Enabled) {

this.\_backGroundImage = new ccui.Scale9Sprite();

this.\_backGroundImage.setPreferredSize(contentSize);

} else this.\_backGroundImage = new cc.Sprite();

this.addProtectedChild(this.\_backGroundImage, ccui.Layout.BACKGROUND\_IMAGE\_ZORDER, -1);

this.\_backGroundImage.setPosition(contentSize.width \* 0.5, contentSize.height \* 0.5);

},

removeBackGroundImage: function () {

if (!this.\_backGroundImage) return;

this.removeProtectedChild(this.\_backGroundImage);

this.\_backGroundImage = null;

this.\_backGroundImageFileName = "";

this.\_backGroundImageTextureSize.width = 0;

this.\_backGroundImageTextureSize.height = 0;

},

setBackGroundColorType: function (type) {

if (this.\_colorType === type) return;

switch (this.\_colorType) {

case ccui.Layout.BG\_COLOR\_NONE:

if (this.\_colorRender) {

this.removeProtectedChild(this.\_colorRender);

this.\_colorRender = null;

}

if (this.\_gradientRender) {

this.removeProtectedChild(this.\_gradientRender);

this.\_gradientRender = null;

}

break;

case ccui.Layout.BG\_COLOR\_SOLID:

if (this.\_colorRender) {

this.removeProtectedChild(this.\_colorRender);

this.\_colorRender = null;

}

break;

case ccui.Layout.BG\_COLOR\_GRADIENT:

if (this.\_gradientRender) {

this.removeProtectedChild(this.\_gradientRender);

this.\_gradientRender = null;

}

break;

default:

break;

}

this.\_colorType = type;

switch (this.\_colorType) {

case ccui.Layout.BG\_COLOR\_NONE:

break;

case ccui.Layout.BG\_COLOR\_SOLID:

this.\_colorRender = new cc.LayerColor();

this.\_colorRender.setContentSize(this.\_contentSize);

this.\_colorRender.setOpacity(this.\_opacity);

this.\_colorRender.setColor(this.\_color);

this.addProtectedChild(this.\_colorRender, ccui.Layout.BACKGROUND\_RENDERER\_ZORDER, -1);

break;

case ccui.Layout.BG\_COLOR\_GRADIENT:

this.\_gradientRender = new cc.LayerGradient(cc.color(255, 0, 0, 255), cc.color(0, 255, 0, 255));

this.\_gradientRender.setContentSize(this.\_contentSize);

this.\_gradientRender.setOpacity(this.\_opacity);

this.\_gradientRender.setStartColor(this.\_startColor);

this.\_gradientRender.setEndColor(this.\_endColor);

this.\_gradientRender.setVector(this.\_alongVector);

this.addProtectedChild(this.\_gradientRender, ccui.Layout.BACKGROUND\_RENDERER\_ZORDER, -1);

break;

default:

break;

}

},

getBackGroundColorType: function () {

return this.\_colorType;

},

setBackGroundColor: function (color, endColor) {

if (!endColor) {

this.\_color.r = color.r;

this.\_color.g = color.g;

this.\_color.b = color.b;

if (this.\_colorRender) this.\_colorRender.setColor(color);

} else {

this.\_startColor.r = color.r;

this.\_startColor.g = color.g;

this.\_startColor.b = color.b;

if (this.\_gradientRender) this.\_gradientRender.setStartColor(color);

this.\_endColor.r = endColor.r;

this.\_endColor.g = endColor.g;

this.\_endColor.b = endColor.b;

if (this.\_gradientRender) this.\_gradientRender.setEndColor(endColor);

}

},

getBackGroundColor: function () {

var tmpColor = this.\_color;

return cc.color(tmpColor.r, tmpColor.g, tmpColor.b, tmpColor.a);

},

getBackGroundStartColor: function () {

var tmpColor = this.\_startColor;

return cc.color(tmpColor.r, tmpColor.g, tmpColor.b, tmpColor.a);

},

getBackGroundEndColor: function () {

var tmpColor = this.\_endColor;

return cc.color(tmpColor.r, tmpColor.g, tmpColor.b, tmpColor.a);

},

setBackGroundColorOpacity: function (opacity) {

this.\_opacity = opacity;

switch (this.\_colorType) {

case ccui.Layout.BG\_COLOR\_NONE:

break;

case ccui.Layout.BG\_COLOR\_SOLID:

this.\_colorRender.setOpacity(opacity);

break;

case ccui.Layout.BG\_COLOR\_GRADIENT:

this.\_gradientRender.setOpacity(opacity);

break;

default:

break;

}

},

getBackGroundColorOpacity: function () {

return this.\_opacity;

},

setBackGroundColorVector: function (vector) {

this.\_alongVector.x = vector.x;

this.\_alongVector.y = vector.y;

if (this.\_gradientRender) {

this.\_gradientRender.setVector(vector);

}

},

getBackGroundColorVector: function () {

return this.\_alongVector;

},

setBackGroundImageColor: function (color) {

this.\_backGroundImageColor.r = color.r;

this.\_backGroundImageColor.g = color.g;

this.\_backGroundImageColor.b = color.b;

this.\_updateBackGroundImageColor();

},

setBackGroundImageOpacity: function (opacity) {

this.\_backGroundImageColor.a = opacity;

this.getBackGroundImageColor();

},

getBackGroundImageColor: function () {

var color = this.\_backGroundImageColor;

return cc.color(color.r, color.g, color.b, color.a);

},

getBackGroundImageOpacity: function () {

return this.\_backGroundImageColor.a;

},

\_updateBackGroundImageColor: function () {

if (this.\_backGroundImage) this.\_backGroundImage.setColor(this.\_backGroundImageColor);

},

getBackGroundImageTextureSize: function () {

return this.\_backGroundImageTextureSize;

},

setLayoutType: function (type) {

this.\_layoutType = type;

var layoutChildrenArray = this.\_children;

var locChild = null;

for (var i = 0; i < layoutChildrenArray.length; i++) {

locChild = layoutChildrenArray[i];

if (locChild instanceof ccui.Widget) this.\_supplyTheLayoutParameterLackToChild(locChild);

}

this.\_doLayoutDirty = true;

},

getLayoutType: function () {

return this.\_layoutType;

},

requestDoLayout: function () {

this.\_doLayoutDirty = true;

},

\_doLayout: function () {

if (!this.\_doLayoutDirty) return;

this.sortAllChildren();

var executant = ccui.getLayoutManager(this.\_layoutType);

if (executant) executant.\_doLayout(this);

this.\_doLayoutDirty = false;

},

\_getLayoutContentSize: function () {

return this.getContentSize();

},

\_getLayoutElements: function () {

return this.getChildren();

},

\_updateBackGroundImageOpacity: function () {

if (this.\_backGroundImage) this.\_backGroundImage.setOpacity(this.\_backGroundImageOpacity);

},

\_updateBackGroundImageRGBA: function () {

if (this.\_backGroundImage) {

this.\_backGroundImage.setColor(this.\_backGroundImageColor);

this.\_backGroundImage.setOpacity(this.\_backGroundImageOpacity);

}

},

\_getLayoutAccumulatedSize: function () {

var children = this.getChildren();

var layoutSize = cc.size(0, 0);

var widgetCount = 0,

locSize;

for (var i = 0, len = children.length; i < len; i++) {

var layout = children[i];

if (null !== layout && layout instanceof ccui.Layout) {

locSize = layout.\_getLayoutAccumulatedSize();

layoutSize.width += locSize.width;

layoutSize.height += locSize.height;

} else {

if (layout instanceof ccui.Widget) {

widgetCount++;

var m = layout.getLayoutParameter().getMargin();

locSize = layout.getContentSize();

layoutSize.width += locSize.width + (m.right + m.left) \* 0.5;

layoutSize.height += locSize.height + (m.top + m.bottom) \* 0.5;

}

}

}

var type = this.getLayoutType();

if (type === ccui.Layout.LINEAR\_HORIZONTAL) layoutSize.height = layoutSize.height - layoutSize.height / widgetCount \* (widgetCount - 1);

if (type === ccui.Layout.LINEAR\_VERTICAL) layoutSize.width = layoutSize.width - layoutSize.width / widgetCount \* (widgetCount - 1);

return layoutSize;

},

\_findNearestChildWidgetIndex: function (direction, baseWidget) {

if (baseWidget == null || baseWidget === this) return this.\_findFirstFocusEnabledWidgetIndex();

var index = 0,

locChildren = this.getChildren();

var count = locChildren.length,

widgetPosition;

var distance = cc.FLT\_MAX,

found = 0;

if (direction === ccui.Widget.LEFT || direction === ccui.Widget.RIGHT || direction === ccui.Widget.DOWN || direction === ccui.Widget.UP) {

widgetPosition = this.\_getWorldCenterPoint(baseWidget);

while (index < count) {

var w = locChildren[index];

if (w && w instanceof ccui.Widget && w.isFocusEnabled()) {

var length = (w instanceof ccui.Layout) ? w.\_calculateNearestDistance(baseWidget) : cc.pLength(cc.pSub(this.\_getWorldCenterPoint(w), widgetPosition));

if (length < distance) {

found = index;

distance = length;

}

}

index++;

}

return found;

}

cc.log("invalid focus direction!");

return 0;

},

\_findFarthestChildWidgetIndex: function (direction, baseWidget) {

if (baseWidget == null || baseWidget === this) return this.\_findFirstFocusEnabledWidgetIndex();

var index = 0,

locChildren = this.getChildren();

var count = locChildren.length;

var distance = -cc.FLT\_MAX,

found = 0;

if (direction === ccui.Widget.LEFT || direction === ccui.Widget.RIGHT || direction === ccui.Widget.DOWN || direction === ccui.Widget.UP) {

var widgetPosition = this.\_getWorldCenterPoint(baseWidget);

while (index < count) {

var w = locChildren[index];

if (w && w instanceof ccui.Widget && w.isFocusEnabled()) {

var length = (w instanceof ccui.Layout) ? w.\_calculateFarthestDistance(baseWidget) : cc.pLength(cc.pSub(this.\_getWorldCenterPoint(w), widgetPosition));

if (length > distance) {

found = index;

distance = length;

}

}

index++;

}

return found;

}

cc.log("invalid focus direction!!!");

return 0;

},

\_calculateNearestDistance: function (baseWidget) {

var distance = cc.FLT\_MAX;

var widgetPosition = this.\_getWorldCenterPoint(baseWidget);

var locChildren = this.\_children;

for (var i = 0, len = locChildren.length; i < len; i++) {

var widget = locChildren[i],

length;

if (widget instanceof ccui.Layout) length = widget.\_calculateNearestDistance(baseWidget);

else {

if (widget instanceof ccui.Widget && widget.isFocusEnabled()) length = cc.pLength(cc.pSub(this.\_getWorldCenterPoint(widget), widgetPosition));

else continue;

}

if (length < distance) distance = length;

}

return distance;

},

\_calculateFarthestDistance: function (baseWidget) {

var distance = -cc.FLT\_MAX;

var widgetPosition = this.\_getWorldCenterPoint(baseWidget);

var locChildren = this.\_children;

for (var i = 0, len = locChildren.length; i < len; i++) {

var layout = locChildren[i];

var length;

if (layout instanceof ccui.Layout) length = layout.\_calculateFarthestDistance(baseWidget);

else {

if (layout instanceof ccui.Widget && layout.isFocusEnabled()) {

var wPosition = this.\_getWorldCenterPoint(layout);

length = cc.pLength(cc.pSub(wPosition, widgetPosition));

} else continue;

}

if (length > distance) distance = length;

}

return distance;

},

\_findProperSearchingFunctor: function (direction, baseWidget) {

if (baseWidget === undefined) return;

var previousWidgetPosition = this.\_getWorldCenterPoint(baseWidget);

var widgetPosition = this.\_getWorldCenterPoint(this.\_findFirstNonLayoutWidget());

if (direction === ccui.Widget.LEFT) {

this.onPassFocusToChild = (previousWidgetPosition.x > widgetPosition.x) ? this.\_findNearestChildWidgetIndex : this.\_findFarthestChildWidgetIndex;

} else if (direction === ccui.Widget.RIGHT) {

this.onPassFocusToChild = (previousWidgetPosition.x > widgetPosition.x) ? this.\_findFarthestChildWidgetIndex : this.\_findNearestChildWidgetIndex;

} else if (direction === ccui.Widget.DOWN) {

this.onPassFocusToChild = (previousWidgetPosition.y > widgetPosition.y) ? this.\_findNearestChildWidgetIndex : this.\_findFarthestChildWidgetIndex;

} else if (direction === ccui.Widget.UP) {

this.onPassFocusToChild = (previousWidgetPosition.y < widgetPosition.y) ? this.\_findNearestChildWidgetIndex : this.\_findFarthestChildWidgetIndex;

} else cc.log("invalid direction!");

},

\_findFirstNonLayoutWidget: function () {

var locChildren = this.\_children;

for (var i = 0, len = locChildren.length; i < len; i++) {

var child = locChildren[i];

if (child instanceof ccui.Layout) {

var widget = child.\_findFirstNonLayoutWidget();

if (widget) return widget;

} else {

if (child instanceof ccui.Widget) return child;

}

}

return null;

},

\_findFirstFocusEnabledWidgetIndex: function () {

var index = 0,

locChildren = this.getChildren();

var count = locChildren.length;

while (index < count) {

var w = locChildren[index];

if (w && w instanceof ccui.Widget && w.isFocusEnabled()) return index;

index++;

}

return 0;

},

\_findFocusEnabledChildWidgetByIndex: function (index) {

var widget = this.\_getChildWidgetByIndex(index);

if (widget) {

if (widget.isFocusEnabled()) return widget;

index = index + 1;

return this.\_findFocusEnabledChildWidgetByIndex(index);

}

return null;

},

\_getWorldCenterPoint: function (widget) {

var widgetSize = widget instanceof ccui.Layout ? widget.\_getLayoutAccumulatedSize() : widget.getContentSize();

return widget.convertToWorldSpace(cc.p(widgetSize.width / 2, widgetSize.height / 2));

},

\_getNextFocusedWidget: function (direction, current) {

var nextWidget = null,

locChildren = this.\_children;

var previousWidgetPos = locChildren.indexOf(current);

previousWidgetPos = previousWidgetPos + 1;

if (previousWidgetPos < locChildren.length) {

nextWidget = this.\_getChildWidgetByIndex(previousWidgetPos);

if (nextWidget) {

if (nextWidget.isFocusEnabled()) {

if (nextWidget instanceof ccui.Layout) {

nextWidget.\_isFocusPassing = true;

return nextWidget.findNextFocusedWidget(direction, nextWidget);

} else {

this.dispatchFocusEvent(current, nextWidget);

return nextWidget;

}

} else return this.\_getNextFocusedWidget(direction, nextWidget);

} else return current;

} else {

if (this.\_loopFocus) {

if (this.\_checkFocusEnabledChild()) {

previousWidgetPos = 0;

nextWidget = this.\_getChildWidgetByIndex(previousWidgetPos);

if (nextWidget.isFocusEnabled()) {

if (nextWidget instanceof ccui.Layout) {

nextWidget.\_isFocusPassing = true;

return nextWidget.findNextFocusedWidget(direction, nextWidget);

} else {

this.dispatchFocusEvent(current, nextWidget);

return nextWidget;

}

} else return this.\_getNextFocusedWidget(direction, nextWidget);

} else return (current instanceof ccui.Layout) ? current : ccui.Widget.\_focusedWidget;

} else {

if (this.\_isLastWidgetInContainer(current, direction)) {

if (this.\_isWidgetAncestorSupportLoopFocus(this, direction)) return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

return (current instanceof ccui.Layout) ? current : ccui.Widget.\_focusedWidget;

} else return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

}

}

},

\_getPreviousFocusedWidget: function (direction, current) {

var nextWidget = null,

locChildren = this.\_children;

var previousWidgetPos = locChildren.indexOf(current);

previousWidgetPos = previousWidgetPos - 1;

if (previousWidgetPos >= 0) {

nextWidget = this.\_getChildWidgetByIndex(previousWidgetPos);

if (nextWidget.isFocusEnabled()) {

if (nextWidget instanceof ccui.Layout) {

nextWidget.\_isFocusPassing = true;

return nextWidget.findNextFocusedWidget(direction, nextWidget);

}

this.dispatchFocusEvent(current, nextWidget);

return nextWidget;

} else return this.\_getPreviousFocusedWidget(direction, nextWidget);

} else {

if (this.\_loopFocus) {

if (this.\_checkFocusEnabledChild()) {

previousWidgetPos = locChildren.length - 1;

nextWidget = this.\_getChildWidgetByIndex(previousWidgetPos);

if (nextWidget.isFocusEnabled()) {

if (nextWidget instanceof ccui.Layout) {

nextWidget.\_isFocusPassing = true;

return nextWidget.findNextFocusedWidget(direction, nextWidget);

} else {

this.dispatchFocusEvent(current, nextWidget);

return nextWidget;

}

} else return this.\_getPreviousFocusedWidget(direction, nextWidget);

} else return (current instanceof ccui.Layout) ? current : ccui.Widget.\_focusedWidget;

} else {

if (this.\_isLastWidgetInContainer(current, direction)) {

if (this.\_isWidgetAncestorSupportLoopFocus(this, direction)) return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

return (current instanceof ccui.Layout) ? current : ccui.Widget.\_focusedWidget;

} else return ccui.Widget.prototype.findNextFocusedWidget.call(this, direction, this);

}

}

},

\_getChildWidgetByIndex: function (index) {

var locChildren = this.\_children;

var size = locChildren.length,

count = 0,

oldIndex = index;

while (index < size) {

var firstChild = locChildren[index];

if (firstChild && firstChild instanceof ccui.Widget) return firstChild;

count++;

index++;

}

var begin = 0;

while (begin < oldIndex) {

var child = locChildren[begin];

if (child && child instanceof ccui.Widget) return child;

count++;

begin++;

}

return null;

},

\_isLastWidgetInContainer: function (widget, direction) {

var parent = widget.getParent();

if (parent == null || !(parent instanceof ccui.Layout)) return true;

var container = parent.getChildren();

var index = container.indexOf(widget);

if (parent.getLayoutType() === ccui.Layout.LINEAR\_HORIZONTAL) {

if (direction === ccui.Widget.LEFT) {

if (index === 0) return this.\_isLastWidgetInContainer(parent, direction);

else return false;

}

if (direction === ccui.Widget.RIGHT) {

if (index === container.length - 1) return this.\_isLastWidgetInContainer(parent, direction);

else return false;

}

if (direction === ccui.Widget.DOWN) return this.\_isLastWidgetInContainer(parent, direction);

if (direction === ccui.Widget.UP) return this.\_isLastWidgetInContainer(parent, direction);

} else if (parent.getLayoutType() === ccui.Layout.LINEAR\_VERTICAL) {

if (direction === ccui.Widget.UP) {

if (index === 0) return this.\_isLastWidgetInContainer(parent, direction);

else return false;

}

if (direction === ccui.Widget.DOWN) {

if (index === container.length - 1) return this.\_isLastWidgetInContainer(parent, direction);

else return false;

}

if (direction === ccui.Widget.LEFT) return this.\_isLastWidgetInContainer(parent, direction);

if (direction === ccui.Widget.RIGHT) return this.\_isLastWidgetInContainer(parent, direction);

} else {

cc.log("invalid layout Type");

return false;

}

},

\_isWidgetAncestorSupportLoopFocus: function (widget, direction) {

var parent = widget.getParent();

if (parent == null || !(parent instanceof ccui.Layout)) return false;

if (parent.isLoopFocus()) {

var layoutType = parent.getLayoutType();

if (layoutType === ccui.Layout.LINEAR\_HORIZONTAL) {

if (direction === ccui.Widget.LEFT || direction === ccui.Widget.RIGHT) return true;

else return this.\_isWidgetAncestorSupportLoopFocus(parent, direction);

}

if (layoutType === ccui.Layout.LINEAR\_VERTICAL) {

if (direction === ccui.Widget.DOWN || direction === ccui.Widget.UP) return true;

else return this.\_isWidgetAncestorSupportLoopFocus(parent, direction);

} else {

cc.assert(0, "invalid layout type");

return false;

}

} else return this.\_isWidgetAncestorSupportLoopFocus(parent, direction);

},

\_passFocusToChild: function (direction, current) {

if (this.\_checkFocusEnabledChild()) {

var previousWidget = ccui.Widget.getCurrentFocusedWidget();

this.\_findProperSearchingFunctor(direction, previousWidget);

var index = this.onPassFocusToChild(direction, previousWidget);

var widget = this.\_getChildWidgetByIndex(index);

if (widget instanceof ccui.Layout) {

widget.\_isFocusPassing = true;

return widget.findNextFocusedWidget(direction, widget);

} else {

this.dispatchFocusEvent(current, widget);

return widget;

}

} else return this;

},

\_checkFocusEnabledChild: function () {

var locChildren = this.\_children;

for (var i = 0, len = locChildren.length; i < len; i++) {

var widget = locChildren[i];

if (widget && widget instanceof ccui.Widget && widget.isFocusEnabled()) return true;

}

return false;

},

getDescription: function () {

return "Layout";

},

\_createCloneInstance: function () {

return new ccui.Layout();

},

\_copyClonedWidgetChildren: function (model) {

ccui.Widget.prototype.\_copyClonedWidgetChildren.call(this, model);

},

\_copySpecialProperties: function (layout) {

if (!(layout instanceof ccui.Layout)) return;

this.setBackGroundImageScale9Enabled(layout.\_backGroundScale9Enabled);

this.setBackGroundImage(layout.\_backGroundImageFileName, layout.\_bgImageTexType);

this.setBackGroundImageCapInsets(layout.\_backGroundImageCapInsets);

this.setBackGroundColorType(layout.\_colorType);

this.setBackGroundColor(layout.\_color);

this.setBackGroundColor(layout.\_startColor, layout.\_endColor);

this.setBackGroundColorOpacity(layout.\_opacity);

this.setBackGroundColorVector(layout.\_alongVector);

this.setLayoutType(layout.\_layoutType);

this.setClippingEnabled(layout.\_clippingEnabled);

this.setClippingType(layout.\_clippingType);

this.\_loopFocus = layout.\_loopFocus;

this.\_\_passFocusToChild = layout.\_\_passFocusToChild;

this.\_isInterceptTouch = layout.\_isInterceptTouch;

},

forceDoLayout: function () {

this.requestDoLayout();

this.\_doLayout();

},

\_createRenderCmd: function () {

if (cc.\_renderType === cc.game.RENDER\_TYPE\_WEBGL) return new ccui.Layout.WebGLRenderCmd(this);

else return new ccui.Layout.CanvasRenderCmd(this);

}

});

var \_p = ccui.Layout.prototype;

\_p.clippingEnabled;

cc.defineGetterSetter(\_p, "clippingEnabled", \_p.isClippingEnabled, \_p.setClippingEnabled);

\_p.clippingType;

cc.defineGetterSetter(\_p, "clippingType", null, \_p.setClippingType);

\_p.layoutType;

cc.defineGetterSetter(\_p, "layoutType", \_p.getLayoutType, \_p.setLayoutType);

\_p = null;

ccui.Layout.create = function () {

return new ccui.Layout();

};

ccui.Layout.BG\_COLOR\_NONE = 0;

ccui.Layout.BG\_COLOR\_SOLID = 1;

ccui.Layout.BG\_COLOR\_GRADIENT = 2;

ccui.Layout.ABSOLUTE = 0;

ccui.Layout.LINEAR\_VERTICAL = 1;

ccui.Layout.LINEAR\_HORIZONTAL = 2;

ccui.Layout.RELATIVE = 3;

ccui.Layout.CLIPPING\_STENCIL = 0;

ccui.Layout.CLIPPING\_SCISSOR = 1;

ccui.Layout.BACKGROUND\_IMAGE\_ZORDER = -1;

ccui.Layout.BACKGROUND\_RENDERER\_ZORDER = -2;

cc.PRIORITY\_NON\_SYSTEM=cc.PRIORITY\_SYSTEM+1;cc.ListEntry=function(prev,next,callback,target,priority,paused,markedForDeletion){this.prev=prev;this.next=next;this.callback=callback;this.target=target;this.priority=priority;this.paused=paused;this.markedForDeletion=markedForDeletion;};cc.HashUpdateEntry=function(list,entry,target,callback,hh){this.list=list;this.entry=entry;this.target=target;this.callback=callback;this.hh=hh;};cc.HashTimerEntry=cc.hashSelectorEntry=function(timers,target,timerIndex,currentTimer,currentTimerSalvaged,paused,hh){var \_t=this;\_t.timers=timers;\_t.target=target;\_t.timerIndex=timerIndex;\_t.currentTimer=currentTimer;\_t.currentTimerSalvaged=currentTimerSalvaged;\_t.paused=paused;\_t.hh=hh;};cc.Timer=cc.Class.extend({\_scheduler:null,\_elapsed:0.0,\_runForever:false,\_useDelay:false,\_timesExecuted:0,\_repeat:0,\_delay:0,\_interval:0.0,getInterval:function(){return this.\_interval;},setInterval:function(interval){this.\_interval=interval;},setupTimerWithInterval:function(seconds,repeat,delay){this.\_elapsed=-1;this.\_interval=seconds;this.\_delay=delay;this.\_useDelay=(this.\_delay>0);this.\_repeat=repeat;this.\_runForever=(this.\_repeat===cc.REPEAT\_FOREVER);},trigger:function(){return 0;},cancel:function(){return 0;},ctor:function(){this.\_scheduler=null;this.\_elapsed=-1;this.\_runForever=false;this.\_useDelay=false;this.\_timesExecuted=0;this.\_repeat=0;this.\_delay=0;this.\_interval=0;},update:function(dt){if(this.\_elapsed===-1){this.\_elapsed=0;this.\_timesExecuted=0;}else{this.\_elapsed+=dt;if(this.\_runForever&&!this.\_useDelay){if(this.\_elapsed>=this.\_interval){this.trigger();this.\_elapsed=0;}}else{if(this.\_useDelay){if(this.\_elapsed>=this.\_delay){this.trigger();this.\_elapsed-=this.\_delay;this.\_timesExecuted+=1;this.\_useDelay=false;}}else{if(this.\_elapsed>=this.\_interval){this.trigger();this.\_elapsed=0;this.\_timesExecuted+=1;}}

if(!this.\_runForever&&this.\_timesExecuted>this.\_repeat)this.cancel();}}}});cc.TimerTargetSelector=cc.Timer.extend({\_target:null,\_selector:null,ctor:function(){this.\_target=null;this.\_selector=null;},initWithSelector:function(scheduler,selector,target,seconds,repeat,delay){this.\_scheduler=scheduler;this.\_target=target;this.\_selector=selector;this.setupTimerWithInterval(seconds,repeat,delay);return true;},getSelector:function(){return this.\_selector;},trigger:function(){if(this.\_target&&this.\_selector){this.\_target.call(this.\_selector,this.\_elapsed);}},cancel:function(){this.\_scheduler.unschedule(this.\_selector,this.\_target);}});cc.TimerTargetCallback=cc.Timer.extend({\_target:null,\_callback:null,\_key:null,ctor:function(){this.\_target=null;this.\_callback=null;},initWithCallback:function(scheduler,callback,target,key,seconds,repeat,delay){this.\_scheduler=scheduler;this.\_target=target;this.\_callback=callback;this.\_key=key;this.setupTimerWithInterval(seconds,repeat,delay);return true;},getCallback:function(){return this.\_callback;},getKey:function(){return this.\_key;},trigger:function(){if(this.\_callback)this.\_callback.call(this.\_target,this.\_elapsed);},cancel:function(){this.\_scheduler.unschedule(this.\_callback,this.\_target);}});cc.Scheduler=cc.Class.extend({\_timeScale:1.0,\_updatesNegList:null,\_updates0List:null,\_updatesPosList:null,\_hashForTimers:null,\_arrayForTimers:null,\_hashForUpdates:null,\_currentTarget:null,\_currentTargetSalvaged:false,\_updateHashLocked:false,ctor:function(){this.\_timeScale=1.0;this.\_updatesNegList=[];this.\_updates0List=[];this.\_updatesPosList=[];this.\_hashForUpdates={};this.\_hashForTimers={};this.\_currentTarget=null;this.\_currentTargetSalvaged=false;this.\_updateHashLocked=false;this.\_arrayForTimers=[];},\_schedulePerFrame:function(callback,target,priority,paused){var hashElement=this.\_hashForUpdates[target.\_\_instanceId];if(hashElement&&hashElement.entry){if(hashElement.entry.priority!==priority){if(this.\_updateHashLocked){cc.log("warning: you CANNOT change update priority in scheduled function");hashElement.entry.markedForDeletion=false;hashElement.entry.paused=paused;return;}else{this.unscheduleUpdate(target);}}else{hashElement.entry.markedForDeletion=false;hashElement.entry.paused=paused;return;}}

if(priority===0){this.\_appendIn(this.\_updates0List,callback,target,paused);}else if(priority<0){this.\_priorityIn(this.\_updatesNegList,callback,target,priority,paused);}else{this.\_priorityIn(this.\_updatesPosList,callback,target,priority,paused);}},\_removeHashElement:function(element){delete this.\_hashForTimers[element.target.\_\_instanceId];cc.arrayRemoveObject(this.\_arrayForTimers,element);element.Timer=null;element.target=null;element=null;},\_removeUpdateFromHash:function(entry){var self=this,element=self.\_hashForUpdates[entry.target.\_\_instanceId];if(element){cc.arrayRemoveObject(element.list,element.entry);delete self.\_hashForUpdates[element.target.\_\_instanceId];element.entry=null;element.target=null;}},\_priorityIn:function(ppList,callback,target,priority,paused){var self=this,listElement=new cc.ListEntry(null,null,callback,target,priority,paused,false);if(!ppList){ppList=[];ppList.push(listElement);}else{var index2Insert=ppList.length-1;for(var i=0;i<=index2Insert;i++){if(priority<ppList[i].priority){index2Insert=i;break;}}

ppList.splice(i,0,listElement);}

self.\_hashForUpdates[target.\_\_instanceId]=new cc.HashUpdateEntry(ppList,listElement,target,null);return ppList;},\_appendIn:function(ppList,callback,target,paused){var self=this,listElement=new cc.ListEntry(null,null,callback,target,0,paused,false);ppList.push(listElement);self.\_hashForUpdates[target.\_\_instanceId]=new cc.HashUpdateEntry(ppList,listElement,target,null,null);},setTimeScale:function(timeScale){this.\_timeScale=timeScale;},getTimeScale:function(){return this.\_timeScale;},update:function(dt){this.\_updateHashLocked=true;if(this.\_timeScale!==1)dt\*=this.\_timeScale;var i,list,len,entry;for(i=0,list=this.\_updatesNegList,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

for(i=0,list=this.\_updates0List,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

for(i=0,list=this.\_updatesPosList,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

var elt,arr=this.\_arrayForTimers;for(i=0;i<arr.length;i++){elt=arr[i];this.\_currentTarget=elt;this.\_currentTargetSalvaged=false;if(!elt.paused){for(elt.timerIndex=0;elt.timerIndex<elt.timers.length;++(elt.timerIndex)){elt.currentTimer=elt.timers[elt.timerIndex];elt.currentTimerSalvaged=false;elt.currentTimer.update(dt);elt.currentTimer=null;}}

if(this.\_currentTargetSalvaged&&this.\_currentTarget.timers.length===0)this.\_removeHashElement(this.\_currentTarget);}

for(i=0,list=this.\_updatesNegList;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

for(i=0,list=this.\_updates0List;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

for(i=0,list=this.\_updatesPosList;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

this.\_updateHashLocked=false;this.\_currentTarget=null;},scheduleCallbackForTarget:function(target,callback\_fn,interval,repeat,delay,paused){this.schedule(callback\_fn,target,interval,repeat,delay,paused,target.\_\_instanceId+"");},schedule:function(callback,target,interval,repeat,delay,paused,key){var isSelector=false;if(typeof callback!=="function"){var selector=callback;isSelector=true;}

if(isSelector===false){if(arguments.length===4||arguments.length===5){key=delay;paused=repeat;delay=0;repeat=cc.REPEAT\_FOREVER;}}else{if(arguments.length===4){paused=repeat;repeat=cc.REPEAT\_FOREVER;delay=0;}}

if(key===undefined){key=target.\_\_instanceId+"";}

cc.assert(target,cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget\_3);var element=this.\_hashForTimers[target.\_\_instanceId];if(!element){element=new cc.HashTimerEntry(null,target,0,null,null,paused,null);this.\_arrayForTimers.push(element);this.\_hashForTimers[target.\_\_instanceId]=element;}else{cc.assert(element.paused===paused,"");}

var timer,i;if(element.timers==null){element.timers=[];}else if(isSelector===false){for(i=0;i<element.timers.length;i++){timer=element.timers[i];if(callback===timer.\_callback){cc.log(cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget,timer.getInterval().toFixed(4),interval.toFixed(4));timer.\_interval=interval;return;}}}else{for(i=0;i<element.timers.length;++i){timer=element.timers[i];if(timer&&selector===timer.getSelector()){cc.log("CCScheduler#scheduleSelector. Selector already scheduled. Updating interval from: %.4f to %.4f",timer.getInterval(),interval);timer.setInterval(interval);return;}}}

if(isSelector===false){timer=new cc.TimerTargetCallback();timer.initWithCallback(this,callback,target,key,interval,repeat,delay);element.timers.push(timer);}else{timer=new cc.TimerTargetSelector();timer.initWithSelector(this,selector,target,interval,repeat,delay);element.timers.push(timer);}},scheduleUpdate:function(target,priority,paused){this.\_schedulePerFrame(function(dt){target.update(dt);},target,priority,paused);},\_getUnscheduleMark:function(key,timer){switch(typeof key){case"number":case"string":return key===timer.getKey();case"function":return key===timer.\_callback;default:return key===timer.getSelector();}},unschedule:function(key,target){if(!target||!key)return;var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){var timers=element.timers;for(var i=0,li=timers.length;i<li;i++){var timer=timers[i];if(this.\_getUnscheduleMark(key,timer)){if((timer===element.currentTimer)&&(!element.currentTimerSalvaged)){element.currentTimerSalvaged=true;}

timers.splice(i,1);if(element.timerIndex>=i){element.timerIndex--;}

if(timers.length===0){if(self.\_currentTarget===element){self.\_currentTargetSalvaged=true;}else{self.\_removeHashElement(element);}}

return;}}}},unscheduleUpdate:function(target){if(target==null)return;var element=this.\_hashForUpdates[target.\_\_instanceId];if(element){if(this.\_updateHashLocked){element.entry.markedForDeletion=true;}else{this.\_removeUpdateFromHash(element.entry);}}},unscheduleAllForTarget:function(target){if(target==null){return;}

var element=this.\_hashForTimers[target.\_\_instanceId];if(element){if(element.timers.indexOf(element.currentTimer)>-1&&(!element.currentTimerSalvaged)){element.currentTimerSalvaged=true;}

element.timers.length=0;if(this.\_currentTarget===element){this.\_currentTargetSalvaged=true;}else{this.\_removeHashElement(element);}}

this.unscheduleUpdate(target);},unscheduleAll:function(){this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},unscheduleAllWithMinPriority:function(minPriority){var i,element,arr=this.\_arrayForTimers;for(i=arr.length-1;i>=0;i--){element=arr[i];this.unscheduleAllForTarget(element.target);}

var entry;var temp\_length=0;if(minPriority<0){for(i=0;i<this.\_updatesNegList.length;){temp\_length=this.\_updatesNegList.length;entry=this.\_updatesNegList[i];if(entry&&entry.priority>=minPriority)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updatesNegList.length)i++;}}

if(minPriority<=0){for(i=0;i<this.\_updates0List.length;){temp\_length=this.\_updates0List.length;entry=this.\_updates0List[i];if(entry)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updates0List.length)i++;}}

for(i=0;i<this.\_updatesPosList.length;){temp\_length=this.\_updatesPosList.length;entry=this.\_updatesPosList[i];if(entry&&entry.priority>=minPriority)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updatesPosList.length)i++;}},isScheduled:function(key,target){cc.assert(key,"Argument key must not be empty");cc.assert(target,"Argument target must be non-nullptr");var element=this.\_hashForUpdates[target.\_\_instanceId];if(!element){return false;}

if(element.timers==null){return false;}else{var timers=element.timers;for(var i=0;i<timers.length;++i){var timer=timers[i];if(key===timer.getKey()){return true;}}

return false;}},pauseAllTargets:function(){return this.pauseAllTargetsWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},pauseAllTargetsWithMinPriority:function(minPriority){var idsWithSelectors=[];var self=this,element,locArrayForTimers=self.\_arrayForTimers;var i,li;for(i=0,li=locArrayForTimers.length;i<li;i++){element=locArrayForTimers[i];if(element){element.paused=true;idsWithSelectors.push(element.target);}}

var entry;if(minPriority<0){for(i=0;i<this.\_updatesNegList.length;i++){entry=this.\_updatesNegList[i];if(entry){if(entry.priority>=minPriority){entry.paused=true;idsWithSelectors.push(entry.target);}}}}

if(minPriority<=0){for(i=0;i<this.\_updates0List.length;i++){entry=this.\_updates0List[i];if(entry){entry.paused=true;idsWithSelectors.push(entry.target);}}}

for(i=0;i<this.\_updatesPosList.length;i++){entry=this.\_updatesPosList[i];if(entry){if(entry.priority>=minPriority){entry.paused=true;idsWithSelectors.push(entry.target);}}}

return idsWithSelectors;},resumeTargets:function(targetsToResume){if(!targetsToResume)return;for(var i=0;i<targetsToResume.length;i++){this.resumeTarget(targetsToResume[i]);}},pauseTarget:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_pauseTarget);var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){element.paused=true;}

var elementUpdate=self.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){elementUpdate.entry.paused=true;}},resumeTarget:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_resumeTarget);var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){element.paused=false;}

var elementUpdate=self.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){elementUpdate.entry.paused=false;}},isTargetPaused:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_isTargetPaused);var element=this.\_hashForTimers[target.\_\_instanceId];if(element){return element.paused;}

var elementUpdate=this.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){return elementUpdate.entry.paused;}

return false;},scheduleUpdateForTarget:function(target,priority,paused){this.scheduleUpdate(target,priority,paused);},unscheduleCallbackForTarget:function(target,callback){this.unschedule(callback,target);},unscheduleUpdateForTarget:function(target){this.unscheduleUpdate(target);},unscheduleAllCallbacksForTarget:function(target){this.unschedule(target.\_\_instanceId+"",target);},unscheduleAllCallbacks:function(){this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},unscheduleAllCallbacksWithMinPriority:function(minPriority){this.unscheduleAllWithMinPriority(minPriority);}});cc.Scheduler.PRIORITY\_SYSTEM=(-2147483647-1);CSSProperties.prototype.\_super;CSSProperties.prototype.ctor;CSSProperties.prototype.Inflate;CSSProperties.prototype.decompress;CSSProperties.prototype.DeviceOrientationEvent;CSSProperties.prototype.DeviceMotionEvent;CSSProperties.prototype.accelerationIncludingGravity;CSSProperties.prototype.gamma;CSSProperties.prototype.beta;CSSProperties.prototype.alpha;var gl=gl||{};CSSProperties.prototype.gl;CSSProperties.prototype.AudioContext;CSSProperties.prototype.webkitAudioContext;CSSProperties.prototype.mozAudioContext;CSSProperties.prototype.createBufferSource;CSSProperties.prototype.createGain;CSSProperties.prototype.createGainNode;CSSProperties.prototype.destination;CSSProperties.prototype.decodeAudioData;CSSProperties.prototype.gain;CSSProperties.prototype.connect;CSSProperties.prototype.playbackState;CSSProperties.prototype.noteGrainOn;CSSProperties.prototype.noteOn;cc.game.addEventListener(cc.game.EVENT\_RENDERER\_INITED,function(){if(cc.\_renderType===cc.game.RENDER\_TYPE\_CANVAS){var \_p=cc.Director.prototype;\_p.getProjection=function(projection){return this.\_projection;};\_p.setProjection=function(projection){this.\_projection=projection;cc.eventManager.dispatchEvent(this.\_eventProjectionChanged);};\_p.setDepthTest=function(){};\_p.setClearColor=function(clearColor){cc.renderer.\_clearColor=clearColor;cc.renderer.\_clearFillStyle='rgb('+clearColor.r+','+clearColor.g+','+clearColor.b+')';};\_p.setOpenGLView=function(openGLView){this.\_winSizeInPoints.width=cc.\_canvas.width;this.\_winSizeInPoints.height=cc.\_canvas.height;this.\_openGLView=openGLView||cc.view;if(cc.eventManager)cc.eventManager.setEnabled(true);};\_p.getVisibleSize=function(){return this.getWinSize();};\_p.getVisibleOrigin=function(){return cc.p(0,0);};}else{cc.Director.\_fpsImage=new Image();cc.Director.\_fpsImage.addEventListener("load",function(){cc.Director.\_fpsImageLoaded=true;});if(cc.\_fpsImage){cc.Director.\_fpsImage.src=cc.\_fpsImage;}}});cc.PI2=Math.PI\*2;cc.DrawingPrimitiveCanvas=cc.Class.extend({\_cacheArray:[],\_renderContext:null,ctor:function(renderContext){this.\_renderContext=renderContext;},drawPoint:function(point,size){if(!size){size=1;}

var locScaleX=cc.view.getScaleX(),locScaleY=cc.view.getScaleY();var newPoint=cc.p(point.x\*locScaleX,point.y\*locScaleY);var ctx=this.\_renderContext.getContext();ctx.beginPath();ctx.arc(newPoint.x,-newPoint.y,size\*locScaleX,0,Math.PI\*2,false);ctx.closePath();ctx.fill();},drawPoints:function(points,numberOfPoints,size){if(points==null)return;if(!size){size=1;}

t.x,-(destPoint.y+destSize.height),destSize.width,destSize.height);break;default:throw new Error("Argument must be non-nil");break;}},drawStar:function(ctx,radius,color){var wrapper=ctx||this.\_renderContext;var context=wrapper.getContext();var colorStr="rgba("+(0|color.r)+","+(0|color.g)+","+(0|color.b);wrapper.setFillStyle(colorStr+",1)");var subRadius=radius/10;context.beginPath();context.moveTo(-radius,radius);context.lineTo(0,subRadius);context.lineTo(radius,radius);context.lineTo(subRadius,0);context.lineTo(radius,-radius);context.lineTo(0,-subRadius);context.lineTo(-radius,-radius);context.lineTo(-subRadius,0);context.lineTo(-radius,radius);context.closePath();context.fill();var rg=context.createRadialGradient(0,0,subRadius,0,0,radius);rg.addColorStop(0,colorStr+", 1)");rg.addColorStop(0.3,colorStr+", 0.8)");rg.addColorStop(1.0,colorStr+", 0.0)");wrapper.setFillStyle(rg);context.beginPath();var startAngle\_1=0;var endAngle\_1=cc.PI2;context.arc(0,0,radius-subRadius,startAngle\_1,endAngle\_1,false);context.closePath();context.fill();},drawColorBall:function(ctx,radius,color){var wrapper=ctx||this.\_renderContext;var context=wrapper.getContext();radius\*=cc.view.getScaleX();var colorStr="rgba("+(0|color.r)+","+(0|color.g)+","+(0|color.b);var subRadius=radius/10;var g1=context.createRadialGradient(0,0,subRadius,0,0,radius);g1.addColorStop(0,colorStr+", 1)");g1.addColorStop(0.3,colorStr+", 0.8)");g1.addColorStop(0.6,colorStr+", 0.4)");g1.addColorStop(1.0,colorStr+", 0.0)");wrapper.setFillStyle(g1);context.beginPath();var startAngle\_1=0;var endAngle\_1=cc.PI2;context.arc(0,0,radius,startAngle\_1,endAngle\_1,false);context.closePath();context.fill();},fillText:function(strText,x,y){this.\_renderContext.getContext().fillText(strText,x,-y);},setDrawColor:function(r,g,b,a){this.\_renderContext.setFillStyle("rgba("+r+","+g+","+b+","+a/255+")");this.\_renderContext.setStrokeStyle("rgba("+r+","+g+","+b+","+a/255+")");},setPointSize:function(pointSize){},setLineWidth:function(width){this.\_renderContext.getContext().lineWidth=width\*cc.view.getScaleX();}});cc.PRIORITY\_NON\_SYSTEM=cc.PRIORITY\_SYSTEM+1;cc.ListEntry=function(prev,next,callback,target,priority,paused,markedForDeletion){this.prev=prev;this.next=next;this.callback=callback;this.target=target;this.priority=priority;this.paused=paused;this.markedForDeletion=markedForDeletion;};cc.HashUpdateEntry=function(list,entry,target,callback,hh){this.list=list;this.entry=entry;this.target=target;this.callback=callback;this.hh=hh;};cc.HashTimerEntry=cc.hashSelectorEntry=function(timers,target,timerIndex,currentTimer,currentTimerSalvaged,paused,hh){var \_t=this;\_t.timers=timers;\_t.target=target;\_t.timerIndex=timerIndex;\_t.currentTimer=currentTimer;\_t.currentTimerSalvaged=currentTimerSalvaged;\_t.paused=paused;\_t.hh=hh;};cc.Timer=cc.Class.extend({\_scheduler:null,\_elapsed:0.0,\_runForever:false,\_useDelay:false,\_timesExecuted:0,\_repeat:0,\_delay:0,\_interval:0.0,getInterval:function(){return this.\_interval;},setInterval:function(interval){this.\_interval=interval;},setupTimerWithInterval:function(seconds,repeat,delay){this.\_elapsed=-1;this.\_interval=seconds;this.\_delay=delay;this.\_useDelay=(this.\_delay>0);this.\_repeat=repeat;this.\_runForever=(this.\_repeat===cc.REPEAT\_FOREVER);},trigger:function(){return 0;},cancel:function(){return 0;},ctor:function(){this.\_scheduler=null;this.\_elapsed=-1;this.\_runForever=false;this.\_useDelay=false;this.\_timesExecuted=0;this.\_repeat=0;this.\_delay=0;this.\_interval=0;},update:function(dt){if(this.\_elapsed===-1){this.\_elapsed=0;this.\_timesExecuted=0;}else{this.\_elapsed+=dt;if(this.\_runForever&&!this.\_useDelay){if(this.\_elapsed>=this.\_interval){this.trigger();this.\_elapsed=0;}}else{if(this.\_useDelay){if(this.\_elapsed>=this.\_delay){this.trigger();this.\_elapsed-=this.\_delay;this.\_timesExecuted+=1;this.\_useDelay=false;}}else{if(this.\_elapsed>=this.\_interval){this.trigger();this.\_elapsed=0;this.\_timesExecuted+=1;}}

if(!this.\_runForever&&this.\_timesExecuted>this.\_repeat)this.cancel();}}}});cc.TimerTargetSelector=cc.Timer.extend({\_target:null,\_selector:null,ctor:function(){this.\_target=null;this.\_selector=null;},initWithSelector:function(scheduler,selector,target,seconds,repeat,delay){this.\_scheduler=scheduler;this.\_target=target;this.\_selector=selector;this.setupTimerWithInterval(seconds,repeat,delay);return true;},getSelector:function(){return this.\_selector;},trigger:function(){if(this.\_target&&this.\_selector){this.\_target.call(this.\_selector,this.\_elapsed);}},cancel:function(){this.\_scheduler.unschedule(this.\_selector,this.\_target);}});cc.TimerTargetCallback=cc.Timer.extend({\_target:null,\_callback:null,\_key:null,ctor:function(){this.\_target=null;this.\_callback=null;},initWithCallback:function(scheduler,callback,target,key,seconds,repeat,delay){this.\_scheduler=scheduler;this.\_target=target;this.\_callback=callback;this.\_key=key;this.setupTimerWithInterval(seconds,repeat,delay);return true;},getCallback:function(){return this.\_callback;},getKey:function(){return this.\_key;},trigger:function(){if(this.\_callback)this.\_callback.call(this.\_target,this.\_elapsed);},cancel:function(){this.\_scheduler.unschedule(this.\_callback,this.\_target);}});cc.Scheduler=cc.Class.extend({\_timeScale:1.0,\_updatesNegList:null,\_updates0List:null,\_updatesPosList:null,\_hashForTimers:null,\_arrayForTimers:null,\_hashForUpdates:null,\_currentTarget:null,\_currentTargetSalvaged:false,\_updateHashLocked:false,ctor:function(){this.\_timeScale=1.0;this.\_updatesNegList=[];this.\_updates0List=[];this.\_updatesPosList=[];this.\_hashForUpdates={};this.\_hashForTimers={};this.\_currentTarget=null;this.\_currentTargetSalvaged=false;this.\_updateHashLocked=false;this.\_arrayForTimers=[];},\_schedulePerFrame:function(callback,target,priority,paused){var hashElement=this.\_hashForUpdates[target.\_\_instanceId];if(hashElement&&hashElement.entry){if(hashElement.entry.priority!==priority){if(this.\_updateHashLocked){cc.log("warning: you CANNOT change update priority in scheduled function");hashElement.entry.markedForDeletion=false;hashElement.entry.paused=paused;return;}else{this.unscheduleUpdate(target);}}else{hashElement.entry.markedForDeletion=false;hashElement.entry.paused=paused;return;}}

if(priority===0){this.\_appendIn(this.\_updates0List,callback,target,paused);}else if(priority<0){this.\_priorityIn(this.\_updatesNegList,callback,target,priority,paused);}else{this.\_priorityIn(this.\_updatesPosList,callback,target,priority,paused);}},\_removeHashElement:function(element){delete this.\_hashForTimers[element.target.\_\_instanceId];cc.arrayRemoveObject(this.\_arrayForTimers,element);element.Timer=null;element.target=null;element=null;},\_removeUpdateFromHash:function(entry){var self=this,element=self.\_hashForUpdates[entry.target.\_\_instanceId];if(element){cc.arrayRemoveObject(element.list,element.entry);delete self.\_hashForUpdates[element.target.\_\_instanceId];element.entry=null;element.target=null;}},\_priorityIn:function(ppList,callback,target,priority,paused){var self=this,listElement=new cc.ListEntry(null,null,callback,target,priority,paused,false);if(!ppList){ppList=[];ppList.push(listElement);}else{var index2Insert=ppList.length-1;for(var i=0;i<=index2Insert;i++){if(priority<ppList[i].priority){index2Insert=i;break;}}

ppList.splice(i,0,listElement);}

self.\_hashForUpdates[target.\_\_instanceId]=new cc.HashUpdateEntry(ppList,listElement,target,null);return ppList;},\_appendIn:function(ppList,callback,target,paused){var self=this,listElement=new cc.ListEntry(null,null,callback,target,0,paused,false);ppList.push(listElement);self.\_hashForUpdates[target.\_\_instanceId]=new cc.HashUpdateEntry(ppList,listElement,target,null,null);},setTimeScale:function(timeScale){this.\_timeScale=timeScale;},getTimeScale:function(){return this.\_timeScale;},update:function(dt){this.\_updateHashLocked=true;if(this.\_timeScale!==1)dt\*=this.\_timeScale;var i,list,len,entry;for(i=0,list=this.\_updatesNegList,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

for(i=0,list=this.\_updates0List,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

for(i=0,list=this.\_updatesPosList,len=list.length;i<len;i++){entry=list[i];if(!entry.paused&&!entry.markedForDeletion)entry.callback(dt);}

var elt,arr=this.\_arrayForTimers;for(i=0;i<arr.length;i++){elt=arr[i];this.\_currentTarget=elt;this.\_currentTargetSalvaged=false;if(!elt.paused){for(elt.timerIndex=0;elt.timerIndex<elt.timers.length;++(elt.timerIndex)){elt.currentTimer=elt.timers[elt.timerIndex];elt.currentTimerSalvaged=false;elt.currentTimer.update(dt);elt.currentTimer=null;}}

if(this.\_currentTargetSalvaged&&this.\_currentTarget.timers.length===0)this.\_removeHashElement(this.\_currentTarget);}

for(i=0,list=this.\_updatesNegList;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

for(i=0,list=this.\_updates0List;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

for(i=0,list=this.\_updatesPosList;i<list.length;){entry=list[i];if(entry.markedForDeletion)this.\_removeUpdateFromHash(entry);else i++;}

this.\_updateHashLocked=false;this.\_currentTarget=null;},scheduleCallbackForTarget:function(target,callback\_fn,interval,repeat,delay,paused){this.schedule(callback\_fn,target,interval,repeat,delay,paused,target.\_\_instanceId+"");},schedule:function(callback,target,interval,repeat,delay,paused,key){var isSelector=false;if(typeof callback!=="function"){var selector=callback;isSelector=true;}

if(isSelector===false){if(arguments.length===4||arguments.length===5){key=delay;paused=repeat;delay=0;repeat=cc.REPEAT\_FOREVER;}}else{if(arguments.length===4){paused=repeat;repeat=cc.REPEAT\_FOREVER;delay=0;}}

if(key===undefined){key=target.\_\_instanceId+"";}

cc.assert(target,cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget\_3);var element=this.\_hashForTimers[target.\_\_instanceId];if(!element){element=new cc.HashTimerEntry(null,target,0,null,null,paused,null);this.\_arrayForTimers.push(element);this.\_hashForTimers[target.\_\_instanceId]=element;}else{cc.assert(element.paused===paused,"");}

var timer,i;if(element.timers==null){element.timers=[];}else if(isSelector===false){for(i=0;i<element.timers.length;i++){timer=element.timers[i];if(callback===timer.\_callback){cc.log(cc.\_LogInfos.Scheduler\_scheduleCallbackForTarget,timer.getInterval().toFixed(4),interval.toFixed(4));timer.\_interval=interval;return;}}}else{for(i=0;i<element.timers.length;++i){timer=element.timers[i];if(timer&&selector===timer.getSelector()){cc.log("CCScheduler#scheduleSelector. Selector already scheduled. Updating interval from: %.4f to %.4f",timer.getInterval(),interval);timer.setInterval(interval);return;}}}

if(isSelector===false){timer=new cc.TimerTargetCallback();timer.initWithCallback(this,callback,target,key,interval,repeat,delay);element.timers.push(timer);}else{timer=new cc.TimerTargetSelector();timer.initWithSelector(this,selector,target,interval,repeat,delay);element.timers.push(timer);}},scheduleUpdate:function(target,priority,paused){this.\_schedulePerFrame(function(dt){target.update(dt);},target,priority,paused);},\_getUnscheduleMark:function(key,timer){switch(typeof key){case"number":case"string":return key===timer.getKey();case"function":return key===timer.\_callback;default:return key===timer.getSelector();}},unschedule:function(key,target){if(!target||!key)return;var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){var timers=element.timers;for(var i=0,li=timers.length;i<li;i++){var timer=timers[i];if(this.\_getUnscheduleMark(key,timer)){if((timer===element.currentTimer)&&(!element.currentTimerSalvaged)){element.currentTimerSalvaged=true;}

timers.splice(i,1);if(element.timerIndex>=i){element.timerIndex--;}

if(timers.length===0){if(self.\_currentTarget===element){self.\_currentTargetSalvaged=true;}else{self.\_removeHashElement(element);}}

return;}}}},unscheduleUpdate:function(target){if(target==null)return;var element=this.\_hashForUpdates[target.\_\_instanceId];if(element){if(this.\_updateHashLocked){element.entry.markedForDeletion=true;}else{this.\_removeUpdateFromHash(element.entry);}}},unscheduleAllForTarget:function(target){if(target==null){return;}

var element=this.\_hashForTimers[target.\_\_instanceId];if(element){if(element.timers.indexOf(element.currentTimer)>-1&&(!element.currentTimerSalvaged)){element.currentTimerSalvaged=true;}

element.timers.length=0;if(this.\_currentTarget===element){this.\_currentTargetSalvaged=true;}else{this.\_removeHashElement(element);}}

this.unscheduleUpdate(target);},unscheduleAll:function(){this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},unscheduleAllWithMinPriority:function(minPriority){var i,element,arr=this.\_arrayForTimers;for(i=arr.length-1;i>=0;i--){element=arr[i];this.unscheduleAllForTarget(element.target);}

var entry;var temp\_length=0;if(minPriority<0){for(i=0;i<this.\_updatesNegList.length;){temp\_length=this.\_updatesNegList.length;entry=this.\_updatesNegList[i];if(entry&&entry.priority>=minPriority)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updatesNegList.length)i++;}}

if(minPriority<=0){for(i=0;i<this.\_updates0List.length;){temp\_length=this.\_updates0List.length;entry=this.\_updates0List[i];if(entry)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updates0List.length)i++;}}

for(i=0;i<this.\_updatesPosList.length;){temp\_length=this.\_updatesPosList.length;entry=this.\_updatesPosList[i];if(entry&&entry.priority>=minPriority)this.unscheduleUpdate(entry.target);if(temp\_length==this.\_updatesPosList.length)i++;}},isScheduled:function(key,target){cc.assert(key,"Argument key must not be empty");cc.assert(target,"Argument target must be non-nullptr");var element=this.\_hashForUpdates[target.\_\_instanceId];if(!element){return false;}

if(element.timers==null){return false;}else{var timers=element.timers;for(var i=0;i<timers.length;++i){var timer=timers[i];if(key===timer.getKey()){return true;}}

return false;}},pauseAllTargets:function(){return this.pauseAllTargetsWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},pauseAllTargetsWithMinPriority:function(minPriority){var idsWithSelectors=[];var self=this,element,locArrayForTimers=self.\_arrayForTimers;var i,li;for(i=0,li=locArrayForTimers.length;i<li;i++){element=locArrayForTimers[i];if(element){element.paused=true;idsWithSelectors.push(element.target);}}

var entry;if(minPriority<0){for(i=0;i<this.\_updatesNegList.length;i++){entry=this.\_updatesNegList[i];if(entry){if(entry.priority>=minPriority){entry.paused=true;idsWithSelectors.push(entry.target);}}}}

if(minPriority<=0){for(i=0;i<this.\_updates0List.length;i++){entry=this.\_updates0List[i];if(entry){entry.paused=true;idsWithSelectors.push(entry.target);}}}

for(i=0;i<this.\_updatesPosList.length;i++){entry=this.\_updatesPosList[i];if(entry){if(entry.priority>=minPriority){entry.paused=true;idsWithSelectors.push(entry.target);}}}

return idsWithSelectors;},resumeTargets:function(targetsToResume){if(!targetsToResume)return;for(var i=0;i<targetsToResume.length;i++){this.resumeTarget(targetsToResume[i]);}},pauseTarget:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_pauseTarget);var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){element.paused=true;}

var elementUpdate=self.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){elementUpdate.entry.paused=true;}},resumeTarget:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_resumeTarget);var self=this,element=self.\_hashForTimers[target.\_\_instanceId];if(element){element.paused=false;}

var elementUpdate=self.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){elementUpdate.entry.paused=false;}},isTargetPaused:function(target){cc.assert(target,cc.\_LogInfos.Scheduler\_isTargetPaused);var element=this.\_hashForTimers[target.\_\_instanceId];if(element){return element.paused;}

var elementUpdate=this.\_hashForUpdates[target.\_\_instanceId];if(elementUpdate){return elementUpdate.entry.paused;}

return false;},scheduleUpdateForTarget:function(target,priority,paused){this.scheduleUpdate(target,priority,paused);},unscheduleCallbackForTarget:function(target,callback){this.unschedule(callback,target);},unscheduleUpdateForTarget:function(target){this.unscheduleUpdate(target);},unscheduleAllCallbacksForTarget:function(target){this.unschedule(target.\_\_instanceId+"",target);},unscheduleAllCallbacks:function(){this.unscheduleAllWithMinPriority(cc.Scheduler.PRIORITY\_SYSTEM);},unscheduleAllCallbacksWithMinPriority:function(minPriority){this.unscheduleAllWithMinPriority(minPriority);}});cc.Scheduler.PRIORITY\_SYSTEM=(-2147483647-1);var initPlayerData=function(){if(gamePlayerData==null){gamePlayerData=new Object();gamePlayerData.soundSwitch=true;gamePlayerData.money=50;gamePlayerData.currentLevel=1;gamePlayerData.levelStarData=[];gamePlayerData.todayData=null;gamePlayerData.gameRecordRemoveMonster=[0,0,0,0,0,0,0,0,0,0];gamePlayerData.gameRemoveMonster=[0,0,0,0,0];gamePlayerData.isCompleteDailyTsk=[false,false,false];gamePlayerData.gameRankingListData=[];gamePlayerData.fristEnterLevelOne=true;gamePlayerData.bestScore=0;gamePlayerData.isJiHuo=1;}

var readData=readPlayerData();for(var key in readData){gamePlayerData[key]=readData[key];}};var Global=Global||{};Global.gameType=1;Global.classicModeWidth=7;Global.classicModeHeight=14;Global.classicModeOffsetX=0;Global.classicModeOffsetY=0;Global.tileMapWidth=90;Global.tileMapHeight=90;Global.monsterWidth=90;Global.monsterHeight=90;Global.monsterFallSpeed=70;Global.createMonsterUpSpeed=5;Global.createMonsterUpTime=8;Global.createMonsterDownTime=8;Global.monsterShakeTimes=50;Global.gameOverData=[];Global.currentMode=1;Global.currentSelectLevel=1;Global.maxSelectLevel=100;Global.monsterDown=false;Global.jiHuoPrice=0;Global.jinBiShuArr=[0,0,0,0];Global.jinBiPriceArr=["0","0","0","0"];Global.jieshuoPage=3;Global.isBaBa=0;Global.warningEffectID=0;Global.monsterSkin=[[res.blue\_leftear\_png,res.blue\_rightear\_png,res.blue\_belly\_png,res.blue\_eyewhites\_png],[res.green\_leftear\_png,res.green\_rightear\_png,res.green\_belly\_png,res.green\_eyewhites\_png],[res.purple\_leftear\_png,res.purple\_rightear\_png,res.purple\_belly\_png,res.purple\_eyewhites\_png],[res.red\_leftear\_png,res.red\_rightear\_png,res.red\_belly\_png,res.red\_eyewhites\_png],[res.yellow\_leftear\_png,res.yellow\_rightear\_png,res.yellow\_belly\_png,res.yellow\_eyewhites\_png]];Global.monsterKingSkin=[[res.big\_blue\_leftear\_png,res.big\_blue\_rightear\_png,res.big\_blue\_belly\_png,res.big\_blue\_number1\_png],[res.big\_green\_leftear\_png,res.big\_green\_rightear\_png,res.big\_green\_belly\_png,res.big\_green\_number1\_png],[res.big\_purple\_leftear\_png,res.big\_purple\_rightear\_png,res.big\_purple\_belly\_png,res.big\_purple\_number1\_png],[res.big\_red\_leftear\_png,res.big\_red\_rightear\_png,res.big\_red\_belly\_png,res.big\_red\_number1\_png],[res.big\_yellow\_leftear\_png,res.big\_yellow\_rightear\_png,res.big\_yellow\_belly\_png,res.big\_yellow\_number1\_png]];Global.music=[res.loginscene\_mp3,res.gamescene\_mp3,res.commonClickDown\_mp3,res.selectClickDown\_mp3,res.pause\_mp3,res.replay\_mp3,res.gamewin\_mp3,res.gamelose\_mp3,res.gameover\_mp3,res.toiletsucker\_mp3,res.dyeingbottle\_mp3,res.lightning\_mp3,res.flamebird\_mp3,res.ice\_mp3,res.box\_mp3,res.synthesis\_mp3,res.threemoreboom\_mp3,res.monsterBoom\_mp3,res.placemonster\_mp3,res.monstershake\_mp3,res.playmonsterthree\_mp3,res.playmonsterfour\_mp3,res.singlemonsterboomTwo\_mp3,res.starlight\_mp3,res.readygo\_mp3,res.warning\_mp3,res.completetask\_mp3,res.completeTarget\_mp3];Global.monsterKingColor=["blue","green","purple","red","yellow"];Global.monsterSpecialEfects=[["starlight",0,8],["flame",0,7],["lightning",0,6],["blackHoleBoom",0,8]];Global.specialMonster=[res.ice\_png,res.timeMonster\_png,res.cage\_png,res.box\_png,res.advancedbox\_png,res.fwordbox\_png,res.fwordadvancedbox\_png];Global.monsterDieEfects=[["blackHole",0,6],["blackSmoke",1,8],["blue",1,7],["green",1,7],["purple",1,7],["red",1,7],["yellow",1,7],["flameBoom",0,4],["starlightBoom",0,5],["timesBoom",0,5],["lightningBoom",0,4],["lightningBoom",5,9],["iceBoom",2,5],["woodenbox",2,7],["ironbox",2,9],["timemonsterboom",0,5],["cageboom",2,9]];Global.gameTargetPicUrl=[res.GameTargetBlue\_png,res.GameTargetGreen\_png,res.GameTargetPurple\_png,res.GameTargetRed\_png,res.GameTargetYellow\_png,res.GameTargetIce\_png,res.GameTargetCage\_png,res.GameTargetBox\_png,res.GameTargetAdvancedBox\_png,res.GameTargetMonsterKing\_png,res.GameTargetScore\_png];Global.dailyTaskMonster=[res.bluemonster\_png,res.greenmonster\_png,res.purplemonster\_png,res.redmonster\_png,res.yellowmonster\_png];Global.gameRecordRemoveMonster=[0,0,0,0,0,0,0,0,0,0];Global.fallCD=[10,9,8,7,6,5,4,3];Global.gameMaps=[[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,5,3,5,0,0],[4,5,3,2,3,5,4],[3,1,1,4,1,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,2,0,0,0,3,3],[2,1,1,5,1,1,2],[4,5,3,2,3,5,4],[9,9,1,4,1,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[8,0,0,0,0,0,8],[8,8,1,5,1,8,8],[4,5,5,2,5,5,4],[1,2,4,3,4,2,1]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,3,6,3,0,0],[2,1,6,1,6,1,2],[4,5,3,6,3,5,4],[3,3,2,5,2,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,11,0,0,0],[0,5,5,1,2,3,0],[4,4,3,2,3,4,4],[3,2,1,5,1,2,3],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,0,0,2,0,0,3],[3,0,0,2,0,0,3],[4,0,5,1,5,0,4],[4,5,3,2,3,5,4],[3,1,1,4,1,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[2,0,0,0,0,0,2],[3,3,0,0,0,3,3],[5,4,1,0,1,4,5],[5,4,3,5,3,4,5],[2,2,1,1,2,2,1]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[1,2,2,5,2,2,1],[3,6,1,3,1,6,3],[3,4,6,5,6,4,3],[4,1,1,4,1,1,4]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,3,0,0,0,4,4],[1,2,1,5,1,1,3],[4,5,6,6,6,5,4],[1,2,3,4,5,4,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[1,2,5,3,1,2,2],[5,1,3,1,2,4,4],[2,2,5,11,3,5,4],[5,3,1,5,1,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,5,3,4,3,5,3],[2,4,1,5,1,4,2]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[2,1,3,1,2,4,4],[3,4,2,2,4,1,5],[2,4,1,5,1,3,2],[4,5,3,2,3,5,3],[6,2,1,6,1,2,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[4,4,1,2,3,2,1],[6,2,3,4,3,1,6],[6,1,2,5,5,1,6],[6,5,3,1,3,5,6],[3,3,2,5,2,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,0,0,0,0,0,6],[3,3,2,5,1,4,2],[4,8,8,8,8,8,4],[3,2,4,5,3,3,2]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,3,1,2,4,2,3],[2,1,4,5,4,3,2],[2,1,6,6,6,4,1]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,3,6,3,6,3,0],[0,6,6,3,6,6,0],[1,5,6,2,6,5,4],[5,3,3,6,3,2,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[8,8,0,0,0,3,5],[8,8,0,2,0,4,2],[2,3,1,5,2,1,2],[1,5,2,4,3,8,8],[2,1,1,4,1,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,2,0,9,0,4,0],[4,2,3,8,3,2,4],[8,8,8,8,8,8,8],[2,3,4,9,3,5,4],[1,2,9,9,9,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,9,9,9,0,0],[0,0,0,0,0,0,0],[1,0,0,0,0,0,4],[2,1,3,8,1,2,3],[3,8,8,8,8,8,4],[2,3,2,8,1,5,5]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,5,0,0,0,5,0],[0,5,0,2,0,5,0],[4,1,1,3,3,2,4],[6,6,4,6,5,6,6],[11,6,1,2,3,6,11],[6,6,1,2,3,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,3,0,0,0,3,0],[2,7,0,0,0,7,2],[5,3,0,3,0,3,5],[8,4,5,2,5,4,8],[7,8,3,2,3,8,7],[8,7,1,8,1,7,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[1,2,5,3,5,2,1],[1,8,8,8,8,8,1],[2,9,9,9,9,9,2],[4,9,7,7,7,9,4],[3,9,9,9,9,9,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,5,0,0,0],[1,0,4,5,4,0,1],[6,0,3,6,3,0,6],[2,6,3,1,3,6,2],[4,5,6,4,6,5,4],[3,2,2,6,2,2,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,0,0,0,0,0,9],[0,9,0,0,0,9,0],[0,0,9,0,9,0,0],[5,0,0,6,0,0,3],[3,1,6,1,6,1,1],[4,6,2,3,2,6,4],[6,4,4,2,4,4,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,2,0,0,0],[0,6,0,8,0,6,0],[0,3,6,8,6,2,0],[1,3,2,6,4,3,1],[2,1,6,8,6,5,2],[2,6,7,8,7,6,2]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,3,1,3,5,4,3],[8,8,8,8,8,8,8],[5,7,8,3,8,7,5],[7,8,3,2,3,8,7],[8,3,2,4,1,4,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,5,4,4,2,4,3],[6,6,6,6,6,6,6],[3,1,1,5,1,1,3],[6,6,6,6,6,6,6],[2,2,1,4,1,4,4]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,9,9,9,9,9,9],[9,9,0,0,0,9,9],[0,0,0,0,0,0,0],[6,6,6,6,3,2,4],[4,4,5,5,4,2,1],[3,1,2,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,10,0,10,0,0],[0,0,0,0,0,0,0],[3,1,2,1,3,4,5],[4,5,2,5,3,4,2],[8,8,8,8,8,8,8],[8,8,8,8,8,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,10,11,10,0,0],[0,0,0,10,0,0,0],[0,0,0,0,0,0,0],[8,8,5,5,2,6,6],[11,8,3,3,1,6,11],[8,8,2,2,4,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,6,6,6,0,0],[3,0,6,3,6,0,3],[5,2,6,2,6,4,4],[3,1,6,6,6,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,3,0,0,0],[0,0,5,5,1,0,0],[0,2,1,1,3,2,0],[8,8,6,4,6,8,8],[8,8,6,7,6,8,8],[8,8,6,6,6,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,0,0,0,0,0,6],[6,6,0,0,0,6,6],[6,6,6,0,6,6,6],[2,1,6,1,6,1,2],[4,5,6,2,6,5,4],[3,3,2,5,2,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,0,9,0,9,0,9],[0,0,0,0,0,0,0],[8,0,0,9,0,0,8],[8,8,0,3,0,8,8],[8,8,8,1,8,8,8],[4,1,1,3,4,2,4],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,11,0,0,0],[0,0,11,8,11,0,0],[8,8,4,3,2,1,5],[2,1,3,1,3,2,4],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,9,0,0,0,9,0],[0,9,0,0,0,9,0],[0,0,9,0,9,0,0],[0,2,0,9,0,2,0],[1,1,5,9,2,1,4],[4,5,3,9,1,5,4],[3,1,1,9,3,1,3]],[[0,0,0,0,0,0,0],[9,0,9,0,9,0,9],[0,9,0,0,0,9,0],[0,9,0,0,0,9,0],[9,0,9,0,9,0,9],[0,0,0,0,0,0,0],[2,1,6,5,6,1,2],[6,6,4,2,3,6,6],[2,6,2,3,4,6,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,10,0,10,0,10,0],[0,0,0,0,0,0,0],[0,0,0,5,0,0,0],[2,3,5,1,4,1,2],[4,5,3,2,3,5,4],[3,4,2,1,2,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,10,0,10,0,0],[0,10,0,0,0,10,0],[8,3,2,2,1,3,8],[8,8,10,5,10,8,8],[4,5,8,2,8,5,4],[4,5,8,1,8,4,2]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,9,0,0,0],[0,0,0,11,0,0,0],[6,6,8,5,8,6,6],[11,2,4,3,1,2,11],[3,4,6,6,6,2,1],[1,2,4,11,5,3,5]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,9,0,1,0,9,9],[9,9,4,4,2,9,9],[9,9,6,3,6,9,9],[6,1,2,4,3,5,6],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,10,10,10,0,0],[0,0,0,3,0,0,0],[4,6,6,1,6,6,4],[1,10,10,2,10,10,2],[3,5,2,4,5,5,3],[3,1,10,10,10,2,5]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,3,4,2,3,4,6],[4,6,3,6,3,6,4],[2,1,6,1,6,1,2],[4,3,3,6,3,2,4],[1,2,6,5,6,5,4]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,8,0,8,0,8,0],[8,3,8,2,8,3,8],[2,8,1,5,1,8,2],[4,5,5,2,5,5,4],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,10,0,0,10,0],[0,10,0,10,0,10,0],[9,0,0,0,0,0,9],[0,3,11,0,11,0,9],[3,2,6,2,6,4,2],[1,6,5,5,4,6,3],[9,9,9,11,9,9,9]],[[0,0,0,0,0,0,0],[9,0,0,0,0,0,9],[9,9,9,9,9,9,9],[9,0,0,0,0,0,9],[0,0,0,0,0,0,0],[4,4,8,8,8,2,4],[3,1,8,7,8,4,2],[2,5,8,5,8,5,2],[3,1,8,7,8,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,2,0,0,0],[0,4,3,5,2,1,0],[0,3,1,3,4,5,0],[6,6,6,6,6,6,6],[6,6,8,8,8,6,6],[6,6,8,8,8,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[2,3,2,1,1,4,5],[3,7,3,4,3,7,7],[6,6,6,6,6,6,6],[6,5,3,6,3,5,6],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,0,0,0,0,0,10],[0,0,0,0,0,0,0],[1,0,3,2,1,0,2],[8,0,8,4,8,0,8],[8,8,8,5,8,8,8],[4,5,3,2,1,5,4],[10,10,10,10,10,10,10]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,5,0,0,0],[0,0,2,3,1,0,0],[0,3,4,5,4,4,0],[0,1,3,1,3,2,0],[0,11,11,11,11,11,0]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,6,6,6,5,3],[0,2,6,3,2,1,5],[0,2,6,6,6,4,4],[4,5,3,1,6,3,2],[3,6,6,6,6,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,2,0,2,0,0],[0,1,6,6,6,1,0],[2,6,6,6,6,6,2],[4,3,10,10,10,5,4],[2,4,10,10,10,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,0,9,0,9,0,9],[0,0,0,6,0,0,0],[0,0,6,7,6,0,0],[0,0,3,6,3,0,0],[2,1,6,1,6,1,2],[4,5,3,6,3,5,4],[3,1,6,9,6,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,4,2,1,3,4,0],[8,5,5,8,5,5,8],[8,8,8,8,8,8,8],[6,6,6,6,6,6,6],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,0,0,4,0,0,10],[3,10,5,10,4,10,4],[2,1,10,0,10,2,3],[6,6,6,6,6,6,6],[6,11,6,11,6,11,6],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,3,0,0,0],[0,0,2,4,1,0,0],[6,9,9,9,1,3,6],[2,3,5,9,5,4,2],[4,5,3,9,3,5,4],[6,1,1,9,9,9,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,9,9,0,9,9,0],[0,9,9,0,9,9,0],[5,9,9,5,9,9,5],[4,6,6,4,6,6,4],[4,6,6,4,6,6,4],[3,6,6,3,6,6,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,4,5,4,0,0],[3,2,9,6,9,2,3],[2,1,9,6,9,1,2],[4,5,9,6,9,5,4],[9,9,9,6,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,5,2,2,0,0],[8,0,3,7,3,0,8],[8,8,1,4,1,8,8],[8,2,5,7,3,5,8],[8,8,2,4,1,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,10,10,10,10],[1,0,0,0,0,0,1],[11,3,0,0,0,2,11],[3,4,0,0,0,1,2],[6,1,5,2,4,2,6],[11,6,2,3,4,6,11]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,6,6,6,0,0],[1,4,5,5,4,3,1],[2,3,9,9,9,2,1],[9,9,6,6,6,9,9],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,6,0,0,0],[0,0,6,9,6,0,0],[0,5,9,9,9,5,0],[6,1,1,9,2,2,6],[4,6,3,9,3,5,4],[2,2,4,4,3,2,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,0,0,0,0,0,0],[4,10,0,0,0,0,4],[5,3,10,6,3,3,2],[2,1,6,10,6,1,2],[6,6,6,6,10,5,3],[2,1,4,5,2,10,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[8,0,0,0,0,0,8],[8,8,8,8,8,8,8],[4,5,8,2,8,5,4],[1,3,5,2,3,3,4]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,0,3,6,4,0,3],[4,0,6,11,6,0,4],[2,6,11,11,11,6,2],[5,6,4,2,1,6,5],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,8,0,0,0,8,0],[3,5,1,2,3,2,1],[3,2,5,3,5,5,4],[4,8,8,4,8,8,4],[1,9,9,4,9,9,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,1,0,0,0],[0,0,2,2,3,0,0],[6,6,6,5,8,8,8],[6,7,3,2,4,7,8],[6,6,6,1,8,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,5,1,5,0,0],[0,5,9,9,9,5,0],[8,8,9,9,9,8,8],[8,8,9,9,9,8,8],[8,8,9,9,9,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,9,9,9,9,0,0],[9,9,9,9,9,9,0],[0,0,4,0,4,0,0],[6,0,3,2,1,0,6],[6,6,1,3,1,6,6],[4,5,3,2,5,5,4],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,9,0,0,0],[0,0,9,0,9,0,0],[0,9,0,1,0,9,0],[0,0,3,2,3,0,0],[8,0,4,4,3,0,8],[8,8,1,5,1,8,8],[6,6,6,11,6,6,6],[6,11,6,11,6,11,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,6,6,6,6,6,6],[9,9,9,9,9,9,9],[6,6,0,0,0,6,6],[1,3,1,5,1,8,8],[4,5,5,2,5,5,4],[3,3,1,3,1,4,5]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,10,0,0,0,0,0],[10,6,10,2,5,3,0],[6,6,6,10,3,5,2],[6,6,6,6,10,4,3],[6,6,6,6,6,10,1],[6,6,6,6,6,6,2]],[[0,0,0,0,0,0,0],[0,0,10,10,0,0,0],[10,10,0,0,0,10,10],[0,0,0,0,0,0,0],[1,0,0,5,0,0,1],[9,2,0,3,0,2,9],[9,9,1,3,1,9,9],[9,9,5,2,5,9,9],[9,9,5,4,3,9,9]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[4,0,0,0,0,0,4],[8,0,0,0,0,0,8],[8,8,8,4,8,8,8],[3,5,4,2,4,5,4],[8,2,8,8,8,1,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,0,10,10,10],[3,0,0,0,0,0,4],[8,3,0,0,0,4,6],[8,8,8,1,1,6,11],[2,5,1,2,6,11,11],[2,3,1,6,11,11,11]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,0,0,0,0,0,0],[6,6,2,0,0,0,0],[6,6,6,6,2,0,0],[6,6,6,6,6,6,3],[5,1,1,8,3,7,2],[1,2,5,8,7,2,3],[8,8,8,8,8,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,0,10,0,10,0,10],[0,10,0,0,0,10,0],[1,0,0,0,0,0,1],[6,2,6,2,6,2,3],[6,3,6,6,6,4,5],[6,5,6,10,6,5,4],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,10,10,10,0,0],[0,0,0,10,0,6,5],[8,0,0,0,0,4,8],[8,8,1,3,1,8,8],[8,5,3,2,5,5,8],[3,2,10,10,10,3,1]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,6],[0,0,0,0,2,6,8],[2,0,0,4,6,8,8],[1,2,4,6,2,4,3],[4,3,6,2,1,8,8],[4,6,5,1,3,3,1],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,10,10,10,0],[10,11,11,11,11,10,0],[10,10,10,10,10,10,0],[0,0,0,0,0,0,0],[4,0,0,0,0,0,0],[3,2,0,0,0,0,2],[3,1,3,0,5,3,1]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,10,10,10,10],[10,10,10,10,10,10,10],[0,0,0,5,0,0,0],[0,0,1,2,1,0,0],[0,2,6,6,6,2,0],[1,1,2,6,2,1,1],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,0,10,10,10],[0,0,10,0,10,0,0],[0,0,10,0,10,0,0],[4,0,0,4,0,0,3],[3,2,1,5,1,1,2],[4,3,3,5,4,5,4],[2,2,1,2,1,3,3]],[[0,0,0,0,0,0,0],[9,9,0,0,0,9,9],[9,9,0,0,0,9,9],[9,9,9,0,9,9,9],[4,0,0,0,0,0,4],[4,0,3,6,3,0,4],[6,1,6,1,6,1,6],[6,5,6,6,6,5,6],[3,3,2,6,2,3,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[8,8,8,8,8,8,8],[9,9,9,9,9,9,9],[0,0,0,0,0,0,0],[2,0,5,0,5,0,1],[4,1,1,5,1,4,2],[9,9,9,9,9,9,9],[8,8,8,8,8,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[11,11,11,11,11,0,0],[10,10,10,10,10,0,0],[0,0,0,0,0,0,0],[4,0,0,0,0,0,4],[1,3,2,5,2,2,5],[2,1,1,2,3,3,4],[11,3,2,4,3,2,11]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,0,10,10,10],[6,6,6,0,6,6,6],[10,6,10,0,10,6,10],[0,6,0,0,0,6,0],[0,4,0,0,0,4,0],[4,3,4,2,2,5,4],[4,1,1,4,1,1,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[8,6,8,0,8,6,8],[6,8,6,0,6,8,6],[8,6,8,5,8,6,8],[2,8,1,3,1,8,3],[8,6,8,4,8,6,8],[2,8,1,2,1,8,3]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,9,9,9,9,9,0],[1,3,1,3,3,1,0],[9,9,9,9,9,9,0],[0,0,3,4,3,0,0],[6,6,6,1,6,6,6],[6,5,6,6,6,5,6],[2,1,2,4,3,2,4]],[[0,0,0,0,0,0,0],[10,0,0,0,0,0,10],[10,0,0,0,0,0,10],[10,0,0,0,0,0,10],[10,0,0,0,0,0,10],[0,0,0,0,0,0,0],[3,4,1,0,1,3,2],[4,5,5,0,3,2,4],[1,2,4,1,3,1,2]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[6,6,6,6,6,6,6],[9,9,9,9,9,9,9],[0,0,0,0,0,0,0],[1,2,3,0,1,3,4],[2,3,1,0,3,2,5],[3,2,1,0,2,4,5],[11,11,11,0,11,11,11]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,5,2,3,10,10],[0,0,10,10,10,0,0],[0,0,1,4,5,0,0],[2,5,2,2,3,5,4],[1,2,1,4,4,1,3]],[[0,0,0,0,0,0,0],[0,6,0,0,0,6,0],[0,10,0,0,0,10,0],[6,0,0,6,0,0,6],[10,0,0,10,0,0,10],[0,0,0,0,0,0,0],[6,2,1,6,2,1,6],[4,10,4,2,10,5,2],[6,2,1,6,1,3,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[9,9,9,9,9,9,9],[8,8,8,8,8,8,8],[9,9,9,9,9,9,9],[0,0,0,4,0,0,0],[2,1,4,1,6,4,2],[8,8,8,8,8,8,8],[9,9,9,9,9,9,9]],[[0,0,0,0,0,0,0],[6,0,0,0,0,0,6],[8,5,0,0,0,5,8],[6,6,0,0,0,6,6],[8,8,5,0,5,8,8],[6,6,6,5,6,6,6],[8,8,8,5,8,8,8],[6,6,6,6,6,6,6],[8,8,8,8,8,8,8]],[[0,0,0,0,0,0,0],[6,6,6,6,6,6,6],[9,9,9,9,9,9,9],[0,0,0,0,0,0,0],[1,0,4,0,1,0,4],[3,0,3,0,2,0,5],[8,11,8,0,8,11,8],[11,11,11,11,11,11,11],[6,6,6,6,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[10,10,10,10,10,10,10],[10,10,10,10,10,10,10],[0,0,0,0,0,0,0],[4,3,3,1,3,3,1],[6,6,6,6,6,6,6],[6,6,6,6,6,6,6],[8,8,8,8,8,8,8]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,6,6,6,6,6,0],[0,6,10,10,10,0,0],[0,6,6,6,6,0,0],[2,6,10,10,6,1,2],[4,6,10,10,6,5,4],[2,6,6,6,6,3,3]],[[0,0,0,0,0,0,0],[0,0,0,10,0,0,0],[0,0,10,0,10,0,0],[0,10,0,0,0,10,0],[10,10,0,0,0,10,10],[2,1,1,0,0,0,0],[6,6,6,3,6,1,2],[4,5,3,6,3,4,3],[3,3,2,5,2,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[1,6,1,0,3,8,1],[6,5,6,1,8,4,8],[5,4,3,3,1,2,5],[6,6,6,5,6,6,6],[4,5,5,6,5,5,4],[6,6,6,8,6,6,6]],[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[3,0,0,0,0,0,3],[6,2,0,0,0,2,6],[11,6,2,0,2,6,11],[6,11,6,0,6,11,6],[8,6,8,4,8,6,8],[8,6,8,2,8,6,8],[6,11,6,1,6,11,6]]];Global.gameTargetData=[[150,30,50,80,5000,0,0,0,0,10,10],[150,30,50,80,0,2,5,8,4,5,8],[150,30,50,80,8000,7,6,0,0,5,8],[150,30,50,80,0,6,4,5,8,3,5],[100,30,50,80,0,10,1,8,7,10,8],[100,30,50,80,0,1,10,5,12,10,10],[100,30,50,80,8500,0,0,2,10,10,10],[100,30,50,80,0,4,10,6,4,10,10],[100,30,50,80,8500,0,0,6,3,10,10],[120,30,50,80,12000,0,0,10,1,10,10],[120,30,50,80,0,4,12,3,15,5,8],[120,30,50,80,9500,0,0,6,3,5,8],[120,30,50,80,0,1,15,6,6,10,8],[150,30,50,80,0,7,5,6,2,10,10],[140,30,50,80,0,9,1,6,3,10,10],[130,30,50,80,9000,0,0,6,9,8,8],[130,30,50,80,9500,0,0,7,8,10,8],[120,30,50,80,0,8,5,7,8,10,8],[120,30,50,80,0,8,3,7,7,10,8],[120,30,50,80,0,0,0,10,2,10,10],[140,30,50,80,10000,0,0,7,7,8,8],[150,30,50,80,0,8,12,7,5,10,10],[110,30,50,80,8000,0,0,6,8,8,8],[130,30,50,80,0,8,6,6,7,10,10],[140,30,50,80,0,7,4,6,9,10,10],[140,30,50,80,0,0,0,7,13,10,10],[140,30,50,80,12000,0,0,6,14,5,6],[140,30,50,80,0,8,11,6,8,5,8],[140,30,50,80,0,9,3,7,14,8,8],[120,30,50,80,0,0,0,10,3,10,10],[150,30,50,80,15000,0,0,6,10,5,8],[150,30,50,80,0,7,12,6,7,10,8],[200,30,50,80,18000,0,0,6,16,10,5],[150,30,50,80,0,7,12,8,12,8,8],[150,30,50,80,0,0,0,10,3,8,8],[120,30,50,80,20000,0,0,8,10,8,8],[130,30,50,80,0,5,12,6,8,8,8],[150,30,50,80,22000,0,0,9,3,8,8],[150,30,50,80,0,9,7,7,10,8,8],[150,30,50,80,0,0,0,10,4,8,8],[150,30,50,80,0,8,12,6,11,8,8],[150,30,50,80,20000,0,0,9,10,8,8],[150,30,50,80,18000,0,0,6,10,5,8],[150,30,50,80,0,7,9,8,7,3,8],[150,30,50,80,0,9,5,10,3,8,8],[150,30,50,80,0,8,11,7,9,8,8],[150,30,50,80,0,7,6,6,15,8,8],[180,30,50,80,22000,0,0,6,17,8,8],[150,30,50,80,0,9,9,7,10,8,8],[150,30,50,80,0,0,0,10,5,8,5],[120,30,45,60,24000,0,0,6,12,8,8],[120,30,45,60,0,6,8,9,6,8,8],[120,30,45,60,0,6,9,8,5,8,8],[120,30,45,60,0,7,10,6,14,8,8],[120,30,45,60,0,9,7,10,3,8,8],[120,30,45,60,0,6,4,8,8,8,8],[120,30,45,60,0,8,12,6,12,8,8],[120,30,45,60,0,8,12,6,4,8,8],[120,30,45,60,28000,0,0,7,12,8,8],[120,30,45,60,30000,0,0,10,4,8,8],[120,30,45,60,0,6,6,8,14,8,8],[120,30,45,60,0,6,6,8,6,8,8],[120,30,45,60,0,9,6,6,7,8,8],[120,30,45,60,32000,0,0,7,11,8,8],[120,30,45,60,0,6,7,10,4,5,8],[120,30,45,60,40000,0,0,0,0,8,8],[120,30,45,60,0,6,7,7,7,8,8],[120,30,45,60,0,8,12,7,12,8,8],[150,30,45,60,0,8,11,6,13,8,8],[120,30,45,60,42000,0,0,10,4,8,8],[120,30,45,60,0,8,7,6,11,8,8],[150,30,50,80,0,9,6,6,19,8,8],[150,30,50,80,0,8,14,9,6,8,8],[150,30,50,80,38000,0,0,7,13,8,8],[150,30,50,80,45000,0,0,10,6,8,8],[150,30,50,80,0,7,9,6,13,8,8],[150,30,50,80,0,9,7,6,17,8,8],[150,30,50,80,0,9,8,7,8,8,8],[150,30,50,80,0,6,13,7,5,8,8],[120,30,45,60,0,10,4,9,14,8,8],[120,30,45,60,0,6,11,9,14,8,8],[120,30,45,60,42000,0,0,9,10,8,8],[150,30,50,80,0,8,14,6,11,8,8],[150,30,50,80,0,7,14,8,14,8,8],[150,30,50,80,50000,10,7,0,0,8,8],[150,30,50,80,0,9,10,6,10,8,8],[150,30,50,80,0,7,18,6,10,8,8],[120,30,45,60,0,8,12,6,11,8,8],[120,30,45,60,50000,9,8,0,0,8,8],[120,30,45,60,55000,0,0,10,6,8,8],[120,30,45,60,52000,9,7,0,0,5,8],[120,30,45,60,0,6,11,9,7,5,8],[120,30,45,60,0,8,21,7,14,5,8],[150,30,50,80,0,6,19,7,19,5,8],[120,30,45,60,58000,10,9,0,0,5,8],[150,30,50,80,0,7,7,9,14,5,8],[120,30,45,60,0,6,18,9,7,5,8],[150,30,50,80,0,9,9,6,7,5,8],[120,30,45,60,0,6,16,7,4,5,8],[120,30,45,60,60000,10,6,0,0,5,8]];var HelloWorldLayer=cc.Layer.extend({sprite:null,ctor:function(){this.\_super();var size=cc.winSize;var helloLabel=new cc.LabelTTF("Hello World","Arial",38);helloLabel.x=size.width/2;helloLabel.y=size.height/2+200;this.addChild(helloLabel,5);this.sprite=new cc.Sprite(res.HelloWorld\_png);this.sprite.attr({x:size.width/2,y:size.height/2});this.addChild(this.sprite,0);return true;}});var HelloWorldScene=cc.Scene.extend({onEnter:function(){this.\_super();var layer=new HelloWorldLayer();this.addChild(layer);}});var res={Button\_Disable\_png:"res/Default/Button\_Disable.png",youlose1:"res/loseLayer/youlose.json",youlose2:"res/loseLayer/youlose.plist",youlose3:"res/loseLayer/youlose.png",youlose4:"res/loseLayer/youlose1.plist",youlose5:"res/loseLayer/youlose1.png",shengli\_json:"res/winLayer/shengli.json",shengli\_png:"res/winLayer/shengli.png",shengli\_plist:"res/winLayer/shengli.plist",shengli1\_png:"res/winLayer/shengli1.png",shengli1\_plist:"res/winLayer/shengli1.plist",gameover\_json:"res/overLayer/gameover.json",gameover\_plist:"res/overLayer/gameover.plist",gameover\_png:"res/overLayer/gameover.png",gameover1\_plist:"res/overLayer/gameover1.plist",gameover1\_png:"res/overLayer/gameover1.png",monster\_json:"res/monster/monster.json",monster\_plist:"res/monster/monster.plist",monster\_png:"res/monster/monster.png",eye\_png:"res/monster/eye.png",blue\_belly\_png:"res/monster/blue/belly.png",blue\_eyewhites\_png:"res/monster/blue/eyewhites.png",blue\_leftear\_png:"res/monster/blue/leftear.png",blue\_rightear\_png:"res/monster/blue/rightear.png",green\_belly\_png:"res/monster/green/belly.png",green\_eyewhites\_png:"res/monster/green/eyewhites.png",green\_leftear\_png:"res/monster/green/leftear.png",green\_rightear\_png:"res/monster/green/rightear.png",purple\_belly\_png:"res/monster/purple/belly.png",purple\_eyewhites\_png:"res/monster/purple/eyewhites.png",purple\_leftear\_png:"res/monster/purple/leftear.png",purple\_rightear\_png:"res/monster/purple/rightear.png",red\_belly\_png:"res/monster/red/belly.png",red\_eyewhites\_png:"res/monster/red/eyewhites.png",red\_leftear\_png:"res/monster/red/leftear.png",red\_rightear\_png:"res/monster/red/rightear.png",yellow\_belly\_png:"res/monster/yellow/belly.png",yellow\_eyewhites\_png:"res/monster/yellow/eyewhites.png",yellow\_leftear\_png:"res/monster/yellow/leftear.png",yellow\_rightear\_png:"res/monster/yellow/rightear.png",monsterKing\_json:"res/monsterKing/monsterKing.json",monsterKing\_plist:"res/monsterKing/monsterKing.plist",monsterKing\_png:"res/monsterKing/monsterKing.png",big\_blue\_belly\_png:"res/monsterKing/blue/belly.png",big\_blue\_leftear\_png:"res/monsterKing/blue/leftear.png",big\_blue\_rightear\_png:"res/monsterKing/blue/rightear.png",big\_blue\_number1\_png:"res/monsterKing/blue/number1.png",big\_blue\_number2\_png:"res/monsterKing/blue/number2.png",big\_blue\_number3\_png:"res/monsterKing/blue/number3.png",big\_blue\_number4\_png:"res/monsterKing/blue/number4.png",big\_blue\_number5\_png:"res/monsterKing/blue/number5.png",big\_green\_belly\_png:"res/monsterKing/green/belly.png",big\_green\_leftear\_png:"res/monsterKing/green/leftear.png",big\_green\_rightear\_png:"res/monsterKing/green/rightear.png",big\_green\_number1\_png:"res/monsterKing/green/number1.png",big\_green\_number2\_png:"res/monsterKing/green/number2.png",big\_green\_number3\_png:"res/monsterKing/green/number3.png",big\_green\_number4\_png:"res/monsterKing/green/number4.png",big\_green\_number5\_png:"res/monsterKing/green/number5.png",big\_purple\_belly\_png:"res/monsterKing/purple/belly.png",big\_purple\_leftear\_png:"res/monsterKing/purple/leftear.png",big\_purple\_rightear\_png:"res/monsterKing/purple/rightear.png",big\_purple\_number1\_png:"res/monsterKing/purple/number1.png",big\_purple\_number2\_png:"res/monsterKing/purple/number2.png",big\_purple\_number3\_png:"res/monsterKing/purple/number3.png",big\_purple\_number4\_png:"res/monsterKing/purple/number4.png",big\_purple\_number5\_png:"res/monsterKing/purple/number5.png",big\_red\_belly\_png:"res/monsterKing/red/belly.png",big\_red\_leftear\_png:"res/monsterKing/red/leftear.png",big\_red\_rightear\_png:"res/monsterKing/red/rightear.png",big\_red\_number1\_png:"res/monsterKing/red/number1.png",big\_red\_number2\_png:"res/monsterKing/red/number2.png",big\_red\_number3\_png:"res/monsterKing/red/number3.png",big\_red\_number4\_png:"res/monsterKing/red/number4.png",big\_red\_number5\_png:"res/monsterKing/red/number5.png",big\_yellow\_belly\_png:"res/monsterKing/yellow/belly.png",big\_yellow\_leftear\_png:"res/monsterKing/yellow/leftear.png",big\_yellow\_rightear\_png:"res/monsterKing/yellow/rightear.png",big\_yellow\_number1\_png:"res/monsterKing/yellow/number1.png",big\_yellow\_number2\_png:"res/monsterKing/yellow/number2.png",big\_yellow\_number3\_png:"res/monsterKing/yellow/number3.png",big\_yellow\_number4\_png:"res/monsterKing/yellow/number4.png",big\_yellow\_number5\_png:"res/monsterKing/yellow/number5.png",SelectScene\_json:"res/SelectScene.json",SelectItemNode\_json:"res/SelectItemNode.json",Back\_png:"res/UI/SelectScene/Back.png",Lock\_png:"res/UI/SelectScene/Lock.png",LockBottom\_png:"res/UI/SelectScene/LockBottom.png",NoPass\_png:"res/UI/SelectScene/NoPass.png",PageViewDark\_png:"res/UI/SelectScene/PageViewDark.png",PageViewLight\_png:"res/UI/SelectScene/PageViewLight.png",Pass\_png:"res/UI/SelectScene/Pass.png",SelectLvBg\_png:"res/UI/SelectScene/SelectLvBg.png",Start\_png:"res/UI/SelectScene/Start.png",StartDark\_png:"res/UI/SelectScene/StartDark.png",StartLight\_png:"res/UI/SelectScene/StartLight.png",StarOne\_png:"res/UI/SelectScene/StarOne.png",StarTwo\_png:"res/UI/SelectScene/StarTwo.png",StarThree\_png:"res/UI/SelectScene/StarThree.png",dailyTaskLayer\_json:"res/dailyTaskLayer.json",bottom\_png:"res/UI/DailyTask/bottom.png",close1\_png:"res/UI/DailyTask/close1.png",close2\_png:"res/UI/DailyTask/close2.png",dailytask\_png:"res/UI/DailyTask/dailytask.png",eliminate\_png:"res/UI/DailyTask/eliminate.png",bottomDown\_png:"res/UI/DailyTask/bottomDown.png",bottomUp\_png:"res/UI/DailyTask/bottomUp.png",gold\_png:"res/UI/DailyTask/gold.png",passLv\_png:"res/UI/DailyTask/passLv.png",received\_png:"res/UI/DailyTask/received.png",recesive\_png:"res/UI/DailyTask/recesive.png",reward\_png:"res/UI/DailyTask/reward.png",taskbotton\_png:"res/UI/DailyTask/taskbotton.png",RankingListLayer\_json:"res/RankingListLayer.json",RankingListNode\_json:"res/RankingListNode.json",Bottom\_png:"res/UI/RankingList/bottom.png",ImperialCrown\_png:"res/UI/RankingList/ImperialCrown.png",LV\_png:"res/UI/RankingList/LV.png",Medals\_png:"res/UI/RankingList/Medals.png",RankingBottom\_png:"res/UI/RankingList/RankingBottom.png",RankingList\_png:"res/UI/RankingList/RankingList.png",score\_png:"res/UI/RankingList/score.png",rankNumber\_plist:"res/UI/RankingList/rankNumber.plist",rankNumber\_png:"res/UI/RankingList/rankNumber.png",GamePauseLayer\_json:"res/GamePauseLayer.json",continue\_png:"res/UI/GamePause/continue.png",gamePauseBottom\_png:"res/UI/GamePause/gamePauseBottom.png",help\_png:"res/UI/GamePause/help.png",home\_png:"res/UI/GamePause/home.png",music1\_png:"res/UI/GamePause/music1.png",music2\_png:"res/UI/GamePause/music2.png",pause\_png:"res/UI/GamePause/pause.png",GameLoseLayer\_json:"res/GameLoseLayer.json",resurrection\_png:"res/UI/GameLose/resurrection.png",GameWinLayer\_json:"res/GameWinLayer.json",GameOVerLayer\_json:"res/GameOVerLayer.json",EndlessGameLayer\_json:"res/EndlessGameLayer.json",LevelGameLayer\_json:"res/LevelGameLayer.json",GameBarBotton\_png:"res/UI/GameScene/GameBarBotton.png",GameBarCentral\_png:"res/UI/GameScene/GameBarCentral.png",GameBarUp\_png:"res/UI/GameScene/GameBarUp.png",GameBigGold\_png:"res/UI/GameScene/GameBigGold.png",GameBotton\_png:"res/UI/GameScene/GameBotton.png",GameChin\_png:"res/UI/GameScene/GameChin.png",GameGoldAdd\_png:"res/UI/GameScene/GameGoldAdd.png",GameGoldBotton\_png:"res/UI/GameScene/GameGoldBotton.png",GameLV\_png:"res/UI/GameScene/GameLV.png",GameScore\_png:"res/UI/GameScene/GameScore.png",GameSmallGold\_png:"res/UI/GameScene/GameSmallGold.png",GameTop2\_png:"res/UI/GameScene/GameTop2.png",Pause\_png:"res/UI/GameScene/Pause.png",pause2\_png:"res/UI/GameScene/pause2.png",GameTarget\_png:"res/UI/GameScene/GameTarget.png",GameTongue\_png:"res/UI/GameScene/GameTongue.png",GameTooth\_png:"res/UI/GameScene/GameTooth.png",GameTooth2\_png:"res/UI/GameScene/GameTooth2.png",GameTop\_png:"res/UI/GameScene/GameTop.png",GameTargetBotton\_png:"res/UI/GameScene/GameTargetBotton.png",GameTargetBlue\_png:"res/UI/GameScene/GameTargetBlue.png",GameTargetGreen\_png:"res/UI/GameScene/GameTargetGreen.png",GameTargetPurple\_png:"res/UI/GameScene/GameTargetPurple.png",GameTargetRed\_png:"res/UI/GameScene/GameTargetRed.png",GameTargetYellow\_png:"res/UI/GameScene/GameTargetYellow.png",GameTargetAdvancedBox\_png:"res/UI/GameScene/GameTargetAdvancedBox.png",GameTargetBox\_png:"res/UI/GameScene/GameTargetBox.png",GameTargetCage\_png:"res/UI/GameScene/GameTargetCage.png",GameTargetIce\_png:"res/UI/GameScene/GameTargetIce.png",GameTargetMonsterKing\_png:"res/UI/GameScene/GameTargetMonsterKing.png",GameTargetScore\_png:"res/UI/GameScene/GameTargetScore.png",gamescenebottom\_png:"res/UI/GameScene/gamescenebottom.png",gamelevelupd\_png:"res/UI/GameScene/gamelevelupd.png",gamelevelupdengji\_png:"res/UI/GameScene/gamelevelupdengji.png",LoginScene\_json:"res/LoginScene.json",ClassicMode\_png:"res/UI/LoginScene/ClassicMode.png",ClassicMode1\_png:"res/UI/LoginScene/ClassicMode1.png",LoginBottonUp\_png:"res/UI/LoginScene/LoginBottonUp.png",eye1\_png:"res/UI/LoginScene/eye1.png",eye2\_png:"res/UI/LoginScene/eye2.png",LevelMode1\_png:"res/UI/LoginScene/LevelMode1.png",LeveMode\_png:"res/UI/LoginScene/LeveMode.png",LoginBotton\_png:"res/UI/LoginScene/LoginBotton.png",LoginMusic\_png:"res/UI/LoginScene/LoginMusic.png",LoginMusic1\_png:"res/UI/LoginScene/LoginMusic1.png",LoginMusic2\_png:"res/UI/LoginScene/LoginMusic2.png",LoginMusic3\_png:"res/UI/LoginScene/LoginMusic3.png",LoginRank\_png:"res/UI/LoginScene/LoginRank.png",LoginRank1\_png:"res/UI/LoginScene/LoginRank1.png",LoginTask\_png:"res/UI/LoginScene/LoginTask.png",LoginTask1\_png:"res/UI/LoginScene/LoginTask1.png",LoginTop\_png:"res/UI/LoginScene/LoginTop.png",MonsterBotton\_png:"res/UI/LoginScene/MonsterBotton.png",monsterPuzzle\_png:"res/UI/LoginScene/monsterPuzzle.png",Tongue\_png:"res/UI/LoginScene/Tongue.png",ToothBotton\_png:"res/UI/LoginScene/ToothBotton.png",blackHole\_plist:"res/dieAnimation/blackHole/blackHole.plist",blackHole\_png:"res/dieAnimation/blackHole/blackHole.png",blackHole1\_plist:"res/dieAnimation/blackHole/blackHole1.plist",blackHole1\_png:"res/dieAnimation/blackHole/blackHole1.png",blackHole2\_plist:"res/dieAnimation/blackHole/blackHole2.plist",blackHole2\_png:"res/dieAnimation/blackHole/blackHole2.png",blackHole3\_plist:"res/dieAnimation/blackHole/blackHole3.plist",blackHole3\_png:"res/dieAnimation/blackHole/blackHole3.png",blackHoleBoom\_plist:"res/dieAnimation/blackHoleBoom/blackHoleBoom.plist",blackHoleBoom\_png:"res/dieAnimation/blackHoleBoom/blackHoleBoom.png",blackSmoke\_plist:"res/dieAnimation/blackSmoke/blackSmoke.plist",blackSmoke\_png:"res/dieAnimation/blackSmoke/blackSmoke.png",blue\_plist:"res/dieAnimation/common/blue/blue.plist",blue\_png:"res/dieAnimation/common/blue/blue.png",green\_plist:"res/dieAnimation/common/green/green.plist",green\_png:"res/dieAnimation/common/green/green.png",purple\_plist:"res/dieAnimation/common/purple/purple.plist",purple\_png:"res/dieAnimation/common/purple/purple.png",red\_plist:"res/dieAnimation/common/red/red.plist",red\_png:"res/dieAnimation/common/red/red.png",yellow\_plist:"res/dieAnimation/common/yellow/yellow.plist",yellow\_png:"res/dieAnimation/common/yellow/yellow.png",flame\_plist:"res/dieAnimation/flame/flame.plist",flame\_png:"res/dieAnimation/flame/flame.png",flameBoom\_plist:"res/dieAnimation/flameBoom/flameBoom.plist",flameBoom\_png:"res/dieAnimation/flameBoom/flameBoom.png",lightning\_plist:"res/dieAnimation/lightning/lightning.plist",lightning\_png:"res/dieAnimation/lightning/lightning.png",starlight\_plist:"res/dieAnimation/starlight/starlight.plist",starlight\_png:"res/dieAnimation/starlight/starlight.png",starlightBoom\_plist:"res/dieAnimation/starlightBoom/starlightBoom.plist",starlightBoom\_png:"res/dieAnimation/starlightBoom/starlightBoom.png",starlightBoom2\_plist:"res/dieAnimation/starlightBoom/starlightBoom2.plist",starlightBoom2\_png:"res/dieAnimation/starlightBoom/starlightBoom2.png",timesBoom\_plist:"res/dieAnimation/timesBoom/timesBoom.plist",timesBoom\_png:"res/dieAnimation/timesBoom/timesBoom.png",dieCommonPlay\_plist:"res/dieAnimation/dieCommon/dieCommonPlay.plist",dieCommonPlay\_png:"res/dieAnimation/dieCommon/dieCommonPlay.png",lightningBoom\_plist:"res/dieAnimation/lightningBoom/lightningBoom.plist",lightningBoom\_png:"res/dieAnimation/lightningBoom/lightningBoom.png",iceBoom\_plist:"res/dieAnimation/iceBoom/iceBoom.plist",iceBoom\_png:"res/dieAnimation/iceBoom/iceBoom.png",woodenbox\_plist:"res/dieAnimation/woodenbox/woodenbox.plist",woodenbox\_png:"res/dieAnimation/woodenbox/woodenbox.png",ironbox\_plist:"res/dieAnimation/ironbox/ironbox.plist",ironbox\_png:"res/dieAnimation/ironbox/ironbox.png",timemonsterboom\_plist:"res/dieAnimation/timemonsterboom/timemonsterboom.plist",timemonsterboom\_png:"res/dieAnimation/timemonsterboom/timemonsterboom.png",cageboom\_plist:"res/dieAnimation/cageboom/cageboom.plist",cageboom\_png:"res/dieAnimation/cageboom/cageboom.png",dyeingbottle\_json:"res/props/dyeingbottle.json",toiletplug\_json:"res/props/toiletplug.json",props\_plist:"res/props/props.plist",props\_png:"res/props/props.png",bluemonster\_png:"res/UI/Icon/bluemonster.png",greenmonster\_png:"res/UI/Icon/greenmonster.png",purplemonster\_png:"res/UI/Icon/purplemonster.png",redmonster\_png:"res/UI/Icon/redmonster.png",yellowmonster\_png:"res/UI/Icon/yellowmonster.png",dyebottle\_png:"res/UI/Icon/dyebottle.png",dyebottlex\_png:"res/UI/Icon/dyebottlex.png",toiletsucker\_png:"res/UI/Icon/toiletsucker.png",toiletsuckerx\_png:"res/UI/Icon/toiletsuckerx.png",advancedbox\_png:"res/specialMonster/advancedbox.png",box\_png:"res/specialMonster/box.png",cage\_png:"res/specialMonster/cage.png",fwordadvancedbox\_png:"res/specialMonster/fwordadvancedbox.png",fwordbox\_png:"res/specialMonster/fwordbox.png",ice\_png:"res/specialMonster/ice.png",timeMonster\_png:"res/specialMonster/timeMonster.png",level\_id\_fnt:"res/level\_id.fnt",level\_id\_0\_png:"res/level\_id\_0.png",level\_star\_fnt:"res/level\_star.fnt",level\_star\_0\_png:"res/level\_star\_0.png",game\_Lv\_fnt:"res/game\_Lv.fnt",game\_Lv\_0\_png:"res/game\_Lv\_0.png",game\_score\_fnt:"res/game\_score.fnt",game\_score\_0\_png:"res/game\_score\_0.png",game\_gold\_fnt:"res/game\_gold.fnt",game\_gold\_0\_png:"res/game\_gold\_0.png",game\_goods\_fnt:"res/game\_goods.fnt",game\_goods\_0\_png:"res/game\_goods\_0.png",game\_time\_fnt:"res/game\_time.fnt",game\_time\_0\_png:"res/game\_time\_0.png",game\_win\_data\_fnt:"res/game\_win\_data.fnt",game\_win\_data\_0\_png:"res/game\_win\_data\_0.png",game\_lose\_data\_fnt:"res/game\_lose\_data.fnt",game\_lose\_data\_0\_png:"res/game\_lose\_data\_0.png",rank\_date\_fnt:"res/rank\_date.fnt",rank\_date\_0\_png:"res/rank\_date\_0.png",rank\_lv\_fnt:"res/rank\_lv.fnt",rank\_lv\_0\_png:"res/rank\_lv\_0.png",rank\_score\_fnt:"res/rank\_score.fnt",rank\_score\_0\_png:"res/rank\_score\_0.png",monsterscore\_fnt:"res/monsterscore.fnt",monsterscore\_0\_png:"res/monsterscore\_0.png",sdkgold\_fnt:"res/sdkgold.fnt",sdkgold\_0\_png:"res/sdkgold\_0.png",gamelevelup\_fnt:"res/gamelevelup.fnt",gamelevelup\_0\_png:"res/gamelevelup\_0.png",mic\_ttf:"res/mic.ttf",comboPic\_plist:"res/comboPic.plist",comboPic\_png:"res/comboPic.png",encourage\_plist:"res/encourage.plist",encourage\_png:"res/encourage.png",loginscene\_mp3:"res/music/loginscene.mp3",gamescene\_mp3:"res/music/gamescene.mp3",commonClickDown\_mp3:"res/music/commonClickDown.mp3",selectClickDown\_mp3:"res/music/selectClickDown.mp3",pause\_mp3:"res/music/pause.mp3",replay\_mp3:"res/music/replay.mp3",gamewin\_mp3:"res/music/gamewin.mp3",gamelose\_mp3:"res/music/gamelose.mp3",gameover\_mp3:"res/music/gameover.mp3",toiletsucker\_mp3:"res/music/toiletsucker.mp3",dyeingbottle\_mp3:"res/music/dyeingbottle.mp3",lightning\_mp3:"res/music/lightning.mp3",flamebird\_mp3:"res/music/flamebird.mp3",ice\_mp3:"res/music/ice.mp3",box\_mp3:"res/music/box.mp3",synthesis\_mp3:"res/music/synthesis.mp3",threemoreboom\_mp3:"res/music/threemoreboom.mp3",monsterBoom\_mp3:"res/music/monsterBoom.mp3",placemonster\_mp3:"res/music/placemonster.mp3",monstershake\_mp3:"res/music/monstershake.mp3",playmonsterthree\_mp3:"res/music/playmonsterthree.mp3",placemonsterfour\_mp3:"res/music/placemonsterfour.mp3",singlemonsterboomTwo\_mp3:"res/music/singlemonsterboomTwo.mp3",starlight\_mp3:"res/music/starlight.mp3",readygo\_mp3:"res/music/readygo.mp3",warning\_mp3:"res/music/warning.mp3",completetask\_mp3:"res/music/completetask.mp3",completeTarget\_mp3:"res/music/completeTarget.mp3",beginnersguide\_png:"res/beginnersguide.png",gameTargetLayer\_json:"res/gameTargetLayer.json",gametargetbottom\_png:"res/UI/GameTarget/gametargetbottom.png",gametargetimg\_png:"res/UI/GameTarget/gametargetimg.png",readygo\_png:"res/UI/GameTarget/readygo.png",TipsLayer\_json:"res/TipsLayer.json",tips1\_png:"res/UI/Tips/tips1.png",tips2\_png:"res/UI/Tips/tips2.png",tips3\_png:"res/UI/Tips/tips3.png",tips4\_png:"res/UI/Tips/tips4.png",tips5\_png:"res/UI/Tips/tips5.png",tips6\_png:"res/UI/Tips/tips6.png",tips7\_png:"res/UI/Tips/tips7.png",tipsbutton\_png:"res/UI/Tips/tipsbutton.png",tooth1\_plist:"res/tooth1.plist",tooth1\_png:"res/tooth1.png",tooth2\_plist:"res/tooth2.plist",tooth2\_png:"res/tooth2.png",ActivationGameLayer\_json:"res/ActivationGameLayer.json",SecondBuyLayer\_json:"res/SecondBuyLayer.json",ShopLayer\_json:"res/ShopLayer.json",PaymentResultLayer\_json:"res/PaymentResultLayer.json",Paymentwaitlayer\_json:"res/Paymentwaitlayer.json",sdk500pic\_png:"res/UI/Sdk/sdk500pic.png",sdkanniu\_png:"res/UI/Sdk/sdkanniu.png",sdkbou\_png:"res/UI/Sdk/sdkbou.png",sdkcgv\_png:"res/UI/Sdk/sdkcgv.png",sdkcgvbotttom2\_png:"res/UI/Sdk/sdkcgvbotttom2.png",sdkcgvsmall\_png:"res/UI/Sdk/sdkcgvsmall.png",sdkcgvsmall1\_png:"res/UI/Sdk/sdkcgvsmall1.png",sdkcgvsmall2\_png:"res/UI/Sdk/sdkcgvsmall2.png",sdkding\_png:"res/UI/Sdk/sdkding.png",sdkding2\_png:"res/UI/Sdk/sdkding2.png",sdkgou\_png:"res/UI/Sdk/sdkgou.png",sdkjinb1\_png:"res/UI/Sdk/sdkjinb1.png",sdkjinb2\_png:"res/UI/Sdk/sdkjinb2.png",sdkjinb3\_png:"res/UI/Sdk/sdkjinb3.png",sdkjinb4\_png:"res/UI/Sdk/sdkjinb4.png",sdkx\_png:"res/UI/Sdk/sdkx.png",timeUP\_png:"res/timeUP.png",GameLevelUpNode\_json:"res/GameLevelUpNode.json"};var g\_resources=[];for(var i in res){g\_resources.push(res[i]);}

var GamePlayScene=GameScene.extend({gameLayer:null,gameLoseLayer:null,gamePauseLayer:null,gameControl:null,upGameLayer:null,gameRecordRemoveMonster:null,gameWinLayer:null,gameOverLayer:null,gameTargetLayer:null,tipsLayer:null,sdkLayer:null,onEnter:function(){this.\_super();this.init();},update:function(data){this.\_super(data);this.gameControl.update(data);},init:function(){this.gameRecordRemoveMonster=[0,0,0,0,0,0,0,0,0,0];initSpriteFrame();this.upGameLayer=new cc.Layer();this.upGameLayer.x=3;this.upGameLayer.y=180;this.addChild(this.upGameLayer,3);this.gameControl=new GameControl();this.gameControl.moveList=[];this.gameControl.removeList=[];this.gameControl.removeSpecialMonsterList=[];this.gameControl.upMonsterList=[];this.gameControl.downMonsterList=[];this.initGameUI();playBgMusic(1);},initGameUI:function(){this.initGameLogicLayer();this.initGameTargetLayer();},initGameTargetLayer:function(){if(Global.currentMode==1){this.gameTargetLayer=new GameTargetLayer();this.addChild(this.gameTargetLayer,4);}},initGameLogicLayer:function(){if(Global.currentMode==1){Global.classicModeHeight=14;this.gameLayer=new LevelGameLayer();}

if(Global.currentMode==2){Global.classicModeHeight=13;this.gameLayer=new EndlessGameLayer();}

this.addChild(this.gameLayer,2);this.gameControl.gameLayer=this.gameLayer;if(Global.currentMode==2){this.gameLayer.initMap();}},gameLose:function(){this.gameLoseLayer=new LoseLayer();this.addChild(this.gameLoseLayer,10);},gamePause:function(){this.gamePauseLayer=new GamePauseLayer();currentScene.addChild(this.gamePauseLayer,10000);},gameWin:function(score,star){this.gameWinLayer=new WinLayer(score,star);this.addChild(this.gameWinLayer,10);},gameOver:function(){this.gameOverLayer=new OverLayer();this.addChild(this.gameOverLayer,10);},gameTips:function(picID){this.tipsLayer=new TipsLayer(picID);this.addChild(this.tipsLayer,10);},createCombo:function(num){if(num>0){var comboNum=new cc.Sprite("#"+"combo"+num+".png");comboNum.x=360;comboNum.y=800+24;this.addChild(comboNum,8);var comboPic=new cc.Sprite("#combo.png");comboPic.x=360+170;comboPic.y=800;this.addChild(comboPic,8);var comboNumCallFun=cc.callFunc(function(){comboNum.removeFromParent();});var comboPicCallFun=cc.callFunc(function(){comboPic.removeFromParent();});comboNum.runAction(cc.sequence(cc.moveBy(0.4,cc.p(0,50)),cc.fadeOut(0.2),comboNumCallFun));comboPic.runAction(cc.sequence(cc.moveBy(0.4,cc.p(0,50)),cc.fadeOut(0.2),comboPicCallFun));}},createEncourage:function(score){var encouragePic=null;if(score<500){cc.log("score < 500");}else if(score>=500&&score<1000){encouragePic=new cc.Sprite("#good.png");}else if(score>=1000&&score<1500){encouragePic=new cc.Sprite("#great.png");}else if(score>=1500&&score<2000){encouragePic=new cc.Sprite("#awesome.png");}else{encouragePic=new cc.Sprite("#verynice.png");}

if(encouragePic!=null){encouragePic.x=360;encouragePic.y=900;this.addChild(encouragePic,8);var encouragePicCallFun=cc.callFunc(function(){encouragePic.removeFromParent();});encouragePic.runAction(cc.sequence(cc.moveBy(0.4,cc.p(0,50)),cc.fadeOut(0.2),encouragePicCallFun));}},createMonsterScore:function(monsterScore,pos){var monsterScoreTxt=new cc.LabelBMFont(monsterScore+"",res.monsterscore\_fnt);monsterScoreTxt.setPosition(pos.x+2,pos.y+200);this.addChild(monsterScoreTxt,8);var monsterScoreCallFun=cc.callFunc(function(){monsterScoreTxt.removeFromParent();});monsterScoreTxt.runAction(cc.sequence(cc.moveBy(0.2,cc.p(0,10)),cc.fadeOut(0.2),monsterScoreCallFun));}});var LoginScene=GameScene.extend({rootNode:null,btnClassicMode:null,btnLevelMode:null,btnRank:null,btnTask:null,btnMusic:null,btnMusicOff:null,gameRankLayer:null,gameTaskLayer:null,saisircode:null,sdkLayer:null,onEnter:function(){this.\_super();engine.gameData.createJsonData();this.init();},init:function(){this.initUI();this.initText();this.initListener();this.initSpriteFrame();this.initTodayTask();},initText:function(){},initTodayTask:function(){var today=new Date();if(gamePlayerData.todayData==null){gamePlayerData.todayData=new Object();gamePlayerData.todayData.getMonth=today.getMonth();gamePlayerData.todayData.getDate=today.getDate();this.initTodayGameRemoveMonster();}else{if(today.getMonth()>=gamePlayerData.todayData.getMonth&&today.getDate()>gamePlayerData.todayData.getDate){gamePlayerData.todayData.getMonth=today.getMonth();gamePlayerData.todayData.getDate=today.getDate();this.initTodayGameRemoveMonster();}}

savePlayerData();},initTodayGameRemoveMonster:function(){gamePlayerData.gameRecordRemoveMonster=[0,0,0,0,0,0,0,0,0,0];gamePlayerData.isCompleteDailyTsk=[false,false,false];var monsterAType=randomNums(0,5);var monsterANum=randomNums(30,51);var monsterBType=randomNums(0,5);if(monsterAType==monsterBType){if(monsterBType!=1){monsterBType=monsterBType-1;}else{monsterBType=monsterBType+1;}}

var monsterBNum=randomNums(30,51);var monsterLevel=gamePlayerData.currentLevel+3;gamePlayerData.gameRemoveMonster[0]=monsterAType;gamePlayerData.gameRemoveMonster[1]=monsterANum;gamePlayerData.gameRemoveMonster[2]=monsterBType;gamePlayerData.gameRemoveMonster[3]=monsterBNum;gamePlayerData.gameRemoveMonster[4]=monsterLevel;savePlayerData();},initUI:function(){initSpriteFrame();this.rootNode=loadCSS(res.LoginScene\_json);this.addChild(this.rootNode);this.btnClassicMode=this.rootNode.Button\_1;this.btnClassicMode.tag=1;this.btnLevelMode=this.rootNode.Button\_2;this.btnLevelMode.tag=2;this.btnRank=this.rootNode.Button\_3;this.btnRank.tag=3;this.btnTask=this.rootNode.Button\_4;this.btnTask.tag=4;this.btnMusic=this.rootNode.Button\_5;this.btnMusic.setCascadeOpacityEnabled(true);this.btnMusic.tag=5;this.btnMusicOff=this.rootNode.Button\_6;this.btnMusicOff.setCascadeOpacityEnabled(true);this.btnMusicOff.tag=6;this.saisircode=this.rootNode.Button\_7;this.saisircode.setPressedActionEnabled(true);this.saisircode.tag=7;this.saisircode.visible=false;playBgMusic(0);this.createEyeAction();this.createTongueAction();this.createMonster();},createEyeAction:function(){var eyeBg=this.rootNode.Image\_3;eyeBg.anchorX=0.5;eyeBg.anchorY=0.4;eyeBg.x=200;eyeBg.y=1080;eyeBg.runAction(cc.sequence(cc.scaleTo(0.08,1,0.7),cc.scaleTo(0.08,1,1),cc.delayTime(0.8)).repeatForever());var eye=this.rootNode.Image\_4;eye.anchorX=0.5;eye.anchorY=0.5;eye.x=165;eye.y=1100;var controlPoints1=[cc.p(235,1100),cc.p(200,1070),cc.p(240,1090)];var controlPoints2=[cc.p(165,1100),cc.p(200,1070),cc.p(160,1090)];var bezierAction1=cc.bezierTo(0.16,controlPoints1);var bezierAction2=cc.bezierTo(0.16,controlPoints2);eye.runAction(cc.sequence(bezierAction1,cc.delayTime(0.8),bezierAction2,cc.delayTime(0.8)).repeatForever());},createTongueAction:function(){var tongue=this.rootNode.Image\_10;tongue.anchorX=0.5;tongue.anchorY=0;tongue.x=360;tongue.y=480;tongue.runAction(cc.sequence(cc.rotateTo(0.3,45),cc.delayTime(2),cc.rotateTo(0.3,-45),cc.delayTime(2)).repeatForever());},createMonster:function(){for(var i=1;i<6;i++){var monster=new SpecialMonsterNode(i);var sprite=this.rootNode.Panel\_1.getChildByName("Sprite\_"+i);monster.x=sprite.x;monster.y=sprite.y;monster.monster.isStop=false;this.rootNode.Panel\_1.addChild(monster);}},initMusicTexture:function(){if(gamePlayerData.soundSwitch){this.btnMusic.visible=true;this.btnMusicOff.visible=false;playBgMusic(0);}else{this.btnMusic.visible=false;this.btnMusicOff.visible=true;stopBgMusic(0);}

savePlayerData();},initListener:function(){this.btnClassicMode.addTouchEventListener(this.btnListenerFun,this.btnClassicMode);this.btnLevelMode.addTouchEventListener(this.btnListenerFun,this.btnLevelMode);this.btnRank.addTouchEventListener(this.btnListenerFun,this.btnRank);this.btnTask.addTouchEventListener(this.btnListenerFun,this.btnTask);this.btnMusic.addTouchEventListener(this.btnListenerFun,this.btnMusic);this.btnMusicOff.addTouchEventListener(this.btnListenerFun,this.btnMusicOff);this.saisircode.addTouchEventListener(this.btnListenerFun,this.saisircode);},btnListenerFun:function(sender,type){switch(sender.tag){case 1:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(3);cc.log("btnClassicMode");if(gamePlayerData.isJiHuo==1){Global.currentMode=2;transitions(beginnersGuideScene);}else{createSDKLayerFun();}

break;}

break;case 2:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(3);cc.log("btnLevelMode");Global.currentMode=1;transitions(SelectScene);break;}

break;case 3:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnRank");currentScene.creatGameRankLayer();break;}

break;case 4:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnTask");currentScene.creatGameTaskLayer();break;}

break;case 5:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnMusic");gamePlayerData.soundSwitch=false;currentScene.initMusicTexture();break;}

break;case 6:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);gamePlayerData.soundSwitch=true;currentScene.initMusicTexture();break;}

break;case 7:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);sendSMSFun();break;}

break;}

savePlayerData();},creatGameRankLayer:function(){this.gameRankLayer=new RankingListLayer();currentScene.addChild(this.gameRankLayer,1111,1111);},creatGameTaskLayer:function(){this.gameTaskLayer=new DailyTaskLayer();currentScene.addChild(this.gameTaskLayer,1111,1111);},initSpriteFrame:function(){cc.spriteFrameCache.addSpriteFrames(res.rankNumber\_plist);}});var beginnersGuideScene=GameScene.extend({beginnersGuideLayer:null,onEnter:function(){this.\_super();this.init();},init:function(){this.beginnersGuideLayer=new cc.Sprite(res.beginnersguide\_png);this.beginnersGuideLayer.x=360;this.beginnersGuideLayer.y=640;this.addChild(this.beginnersGuideLayer);var callFunAction=cc.callFunc(function(){transitions(GamePlayScene);});this.runAction(cc.sequence(cc.delayTime(1.5),callFunAction));}});var angleUnit=Math.PI/180;function randomNums(Min,Max){var Range=Max-Min;var Rand=Math.random();return(Min+Math.floor(Rand\*Range));}

var shakeAction=function(){var shakeDown=cc.moveBy(0.08,cc.p(0,-7));var shakeUp=cc.moveBy(0.24,cc.p(0,7));return[shakeUp,shakeDown];};var monsterShakeAction=function(rowcol){var lx=currentScene.gameLayer.tileMapPos[rowcol[0]][rowcol[1]].x;var ly=currentScene.gameLayer.tileMapPos[rowcol[0]][rowcol[1]].y;var shakeDown=cc.moveTo(0.03,cc.p(lx-1,ly-5));var shakeUp=cc.moveTo(0.04,cc.p(lx+5,ly+2));var shakeLeft=cc.moveTo(0.02,cc.p(lx-3,ly+1));var shakeRight=cc.moveTo(0.01,cc.p(lx+2,ly-4));var shake1=cc.moveTo(0.03,cc.p(lx+0,ly+2));var shake2=cc.moveTo(0.02,cc.p(lx+4,ly-3));var shake3=cc.moveTo(0.01,cc.p(lx-3,ly-2));var shake4=cc.moveTo(0.03,cc.p(lx+2,ly-5));var shake5=cc.moveTo(0.02,cc.p(lx,ly));return[shakeUp,shakeDown,shakeLeft,shakeRight,shake1,shake2,shake3,shake4,shake5];};function createMonsterData(monster,j,i){var data=[];data.push(monster);data.push(j);data.push(i);return data;}

function searchMaxColFromArr(arr){if(arr==null)return;var length=arr.length;var maxCol=0;var space=0;for(var i=0;i<length;i++){if(arr[i]>space){space=arr[i];maxCol=i;}}

return maxCol;}

function getIDByName(name){}

function playBgMusic(id){if(gamePlayerData.soundSwitch){cc.audioEngine.playMusic(Global.music[id],true);}}

function stopBgMusic(id){cc.audioEngine.stopMusic(Global.music[id]);}

function playEffect(id){if(gamePlayerData.soundSwitch){cc.audioEngine.playEffect(Global.music[id]);}}

function playWarningEffect(id){if(gamePlayerData.soundSwitch){if(id==25){Global.warningEffectID=cc.audioEngine.playEffect(Global.music[id],true);}else{cc.audioEngine.playEffect(Global.music[id],true);}}}

function stopWarningEffect(){cc.audioEngine.stopEffect(Global.warningEffectID);}

function initSpriteFrame(){cc.spriteFrameCache.addSpriteFrames(res.monster\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackHole\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackHole1\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackHole2\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackHole3\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackHoleBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.blackSmoke\_plist);cc.spriteFrameCache.addSpriteFrames(res.blue\_plist);cc.spriteFrameCache.addSpriteFrames(res.green\_plist);cc.spriteFrameCache.addSpriteFrames(res.purple\_plist);cc.spriteFrameCache.addSpriteFrames(res.red\_plist);cc.spriteFrameCache.addSpriteFrames(res.yellow\_plist);cc.spriteFrameCache.addSpriteFrames(res.flame\_plist);cc.spriteFrameCache.addSpriteFrames(res.flameBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.lightning\_plist);cc.spriteFrameCache.addSpriteFrames(res.lightningBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.starlight\_plist);cc.spriteFrameCache.addSpriteFrames(res.starlightBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.starlightBoom2\_plist);cc.spriteFrameCache.addSpriteFrames(res.timesBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.dieCommonPlay\_plist);cc.spriteFrameCache.addSpriteFrames(res.props\_plist);cc.spriteFrameCache.addSpriteFrames(res.iceBoom\_plist);cc.spriteFrameCache.addSpriteFrames(res.woodenbox\_plist);cc.spriteFrameCache.addSpriteFrames(res.ironbox\_plist);cc.spriteFrameCache.addSpriteFrames(res.timemonsterboom\_plist);cc.spriteFrameCache.addSpriteFrames(res.cageboom\_plist);cc.spriteFrameCache.addSpriteFrames(res.monsterKing\_plist);cc.spriteFrameCache.addSpriteFrames(res.comboPic\_plist);cc.spriteFrameCache.addSpriteFrames(res.encourage\_plist);cc.spriteFrameCache.addSpriteFrames(res.rankNumber\_plist);cc.spriteFrameCache.addSpriteFrames(res.tooth1\_plist);cc.spriteFrameCache.addSpriteFrames(res.tooth2\_plist);}

function addGameRankingListData(lv,date,score){var data=new Object();data.lv=lv;data.date=date;data.score=score;gamePlayerData.gameRankingListData.push(data);}

function sortGameRankingListData(){gamePlayerData.gameRankingListData.sort(function(a,b){return b.score-a.score});popGameRankingListData();}

function popGameRankingListData(){var length=gamePlayerData.gameRankingListData.length;if(length<=10){cc.log("gamePlayerData.gameRankingListData.length <= 10");}else{gamePlayerData.gameRankingListData.pop();}}

var TipsLayer=cc.Layer.extend({rootNode:null,ctor:function(picID){this.\_super();this.init(picID);},init:function(picID){this.initUI(picID);cc.director.pause();},initUI:function(picID){this.rootNode=loadCSS(res.TipsLayer\_json);this.addChild(this.rootNode);this.rootNode.Sprite\_1.initWithFile("res/UI/Tips/tips"+picID+".png");this.rootNode.Button\_1.setPressedActionEnabled(true);this.rootNode.Button\_1.addTouchEventListener(this.btnDetermineFun);if(picID==3){this.rootNode.Button\_1.y=100;}},btnDetermineFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("Determine");if(cc.director.isPaused()){cc.director.resume();}

currentScene.tipsLayer.removeFromParent();break;}}});var OverLayer=cc.Layer.extend({rootNode:null,home:null,homePic:null,replay:null,replayPic:null,gameover:null,ctor:function(){this.\_super();this.init();},init:function(){this.initUI();this.initText();this.initListener();},initUI:function(){this.rootNode=loadCSS(res.GameOVerLayer\_json);this.addChild(this.rootNode);var animationPanel=this.rootNode.Panel\_3;cc.spriteFrameCache.addSpriteFrames(res.gameover\_plist);cc.spriteFrameCache.addSpriteFrames(res.gameover1\_plist);multipleAnimation(res.gameover\_json,res.gameover\_json);this.gameover=new GameSkeletalAnimation();var data=skeletalAnimation[res.gameover\_json+"|gameover"];data.playCount=1;this.gameover.initialize(data);this.gameover.setFrameIntervalTime(24);this.gameover.isRemoveFromComplete=false;animationPanel.addChild(this.gameover);this.gameover.x=80;this.gameover.y=720;this.home=new GameSkeletalAnimation();var dataHome=skeletalAnimation[res.gameover\_json+"|gamehome"];dataHome.playCount=1;this.home.initialize(dataHome);this.home.setFrameIntervalTime(24);this.home.isRemoveFromComplete=false;animationPanel.addChild(this.home);this.home.x=260;this.home.y=420;this.replay=new GameSkeletalAnimation();var dataReplay=skeletalAnimation[res.gameover\_json+"|gamereplay"];dataReplay.playCount=1;this.replay.initialize(dataReplay);this.replay.setFrameIntervalTime(24);this.replay.isRemoveFromComplete=false;animationPanel.addChild(this.replay);this.replay.x=460;this.replay.y=420;this.homePic=this.home.getBimapByUrl("#png19.png");this.replayPic=this.replay.getBimapByUrl("#png20.png");playEffect(8);},initText:function(){this.rootNode.BitmapFontLabel\_1.setString(Global.gameOverData[0]+"");this.rootNode.BitmapFontLabel\_2.setString(Global.gameOverData[1]+"");this.rootNode.BitmapFontLabel\_3.setString(Global.gameOverData[2]+"");},initListener:function(){engine.gameEvent.createMouseDown(this.homePic,this.btnHomeFun);engine.gameEvent.createMouseDown(this.replayPic,this.btnReplayFun);},btnHomeFun:function(){playEffect(2);cc.log("btnHomeFun");createCCSButtonScaling(this);transitions(LoginScene);},btnReplayFun:function(){playEffect(5);cc.log("btnReplayFun");createCCSButtonScaling(this);transitions(GamePlayScene);}

var LoseLayer=cc.Layer.extend({rootNode:null,resurrectionImg:null,resurrectionImgPic:null,goldText:null,home:null,homePic:null,replay:null,replayPic:null,gamelose:null,ctor:function(){this.\_super();this.init();},init:function(){this.initUI();var self=this;var callFunAction=cc.callFunc(function(){self.initText();self.initListener();});this.runAction(cc.sequence(cc.delayTime(1),callFunAction));},initUI:function(){this.rootNode=loadCSS(res.GameLoseLayer\_json);this.addChild(this.rootNode);this.rootNode.BitmapFontLabel\_1.visibie=false;var animationPanel=this.rootNode.Panel\_2;cc.spriteFrameCache.addSpriteFrames(res.youlose2);cc.spriteFrameCache.addSpriteFrames(res.youlose4);multipleAnimation(res.youlose1,res.youlose1);this.gamelose=new GameSkeletalAnimation();var data=skeletalAnimation[res.youlose1+"|youlose"];data.playCount=1;this.gamelose.initialize(data);this.gamelose.setFrameIntervalTime(24);this.gamelose.isRemoveFromComplete=false;animationPanel.addChild(this.gamelose);this.gamelose.x=80;this.gamelose.y=720;this.home=new GameSkeletalAnimation();var dataHome=skeletalAnimation[res.youlose1+"|gamehome"];dataHome.playCount=1;this.home.initialize(dataHome);this.home.setFrameIntervalTime(24);this.home.isRemoveFromComplete=false;animationPanel.addChild(this.home);this.home.x=260;this.home.y=420;this.replay=new GameSkeletalAnimation();var dataReplay=skeletalAnimation[res.youlose1+"|gamereplay"];dataReplay.playCount=1;this.replay.initialize(dataReplay);this.replay.setFrameIntervalTime(24);this.replay.isRemoveFromComplete=false;animationPanel.addChild(this.replay);this.replay.x=460;this.replay.y=420;this.resurrectionImg=new GameSkeletalAnimation();var dataResurrectionImg=skeletalAnimation[res.youlose1+"|gameresurrectionImg"];dataResurrectionImg.playCount=1;this.resurrectionImg.initialize(dataResurrectionImg);this.resurrectionImg.setFrameIntervalTime(24);this.resurrectionImg.isRemoveFromComplete=false;animationPanel.addChild(this.resurrectionImg);this.resurrectionImg.x=360;this.resurrectionImg.y=540;this.homePic=this.home.getBimapByUrl("#youlosepng19.png");this.replayPic=this.replay.getBimapByUrl("#youlosepng20.png");this.resurrectionImgPic=this.resurrectionImg.getBimapByUrl("#youlosepng22.png");playEffect(7);},initText:function(){this.rootNode.BitmapFontLabel\_1.visibie=true;},initListener:function(){engine.gameEvent.createMouseDown(this.homePic,this.btnHomeFun);engine.gameEvent.createMouseDown(this.replayPic,this.btnReplayFun);engine.gameEvent.createMouseDown(this.resurrectionImgPic,this.btnResurrectionFun);},showElement:function(obj){},btnHomeFun:function(){playEffect(2);cc.log("btnHomeFun");createCCSButtonScaling(this);transitions(SelectScene);},btnReplayFun:function(){playEffect(5);cc.log("btnReplayFun");createCCSButtonScaling(this);transitions(GamePlayScene);},btnResurrectionFun:function(){playEffect(2);cc.log("btnResurrectionFun");createCCSButtonScaling(this);if(gamePlayerData.money>=300){gamePlayerData.money-=300;if(currentScene.gameLayer.isFullMonster){currentScene.gameLayer.isFullMonster=false;var grid=[];for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<7;j++){var monster=currentScene.gameLayer.tileMap.gridArr[i][j];if(monster!=null&&monster.isMonster){grid=monster.rowcol;break;}}}

currentScene.gameLayer.useToiletPlug([grid[1],grid[0]]);var callFunI=cc.callFunc(function(){currentScene.gameLayer.startScheduleCheckout();});currentScene.gameLayer.runAction(cc.sequence(cc.delayTime(3),callFunI));}else{currentScene.gameLayer.startScheduleCheckout();}

currentScene.gameLayer.countTime+=30;currentScene.gameLayer.isGameOver=false;currentScene.gameLoseLayer.removeFromParent();currentScene.gameLoseLayer=null;}

currentScene.gameLayer.refreshCurrentMoney();savePlayerData();}});});var Mpos=new cc.p(0,0);var levelGameLayerTargetOffset=120;var LevelGameLayer=cc.Layer.extend({rootNode:null,btnPause:null,btnAddGold:null,btnToiletSucker:null,btnDyeBottle:null,panel:null,touchPanel:null,tileMap:null,tileMapx:null,tileMapPos:null,tileMapUp:null,tileMapDown:null,tempMonsterArr:null,tempMonsterArrs:null,removeRows:null,touchMonster:null,isCreateMonsterDown:false,isCreateCompleted:false,tempColsArr:null,isClearMonster:false,isTouchDown:false,touchPosition:null,gameControl:null,touchPosInBox:null,frontGrid:null,currentGrid:null,isGenerateMonster:false,connectMonsterArrs:null,isTouchInBox:false,isSelectDyeBottle:false,isDyeing:false,isSelectToiletSucker:false,isPlayToilet:false,isBanFall:false,countTime:0,score:0,gameStarOne:0,gameStarTwo:0,gameStarThree:0,gameTargetScore:0,gameTargetOneID:0,gameTargetOneNum:0,gameTargetTwoID:0,gameTargetTwoNum:0,gameInitFallCD:0,gameFallCD:0,gameTarget:null,isGameWin:false,isGameOver:false,isPause:false,isTouchOnMouseUp:false,isShowTipsTwo:false,isShowTipsThree:false,isShowGameAlarm:false,fallToothTime:null,isCanAddTimeMonster:false,addTimeMonsterID:0,isFullMonster:false,isTimeUP:false,isPlayWarningEffect:false,isPlayCompleteTargetOneEffect:false,isPlayCompleteTargetTwoEffect:false,isCanTouchDown:true,isCanTouchDownCD:0,ctor:function(){this.\_super();this.init();this.scheduleUpdate();this.scheduleOnce(this.scheduleFallMonster,this.gameInitFallCD);},scheduleFun:function(){this.refreshTime();this.schedule(this.updateCheckout,1);},update:function(dt){if(!this.isCanTouchDown){this.isCanTouchDownCD+=dt;if(this.isCanTouchDownCD>0.06){this.isCanTouchDownCD=0;this.isCanTouchDown=true;}}},scheduleFallMonster:function(){this.schedule(this.fallMonster,this.gameFallCD);},startScheduleCheckout:function(){this.schedule(this.updateCheckout,1);},stopScheduleCheckout:function(){this.unschedule(this.updateCheckout);},init:function(){if(Global.currentSelectLevel==1){this.isShowTipsTwo=false;this.isShowTipsThree=false;}

this.initGameTargetData();this.initUI();this.initData();this.initText();this.initListener();},initGameTargetData:function(){this.gameTarget=[];this.countTime=Global.gameTargetData[Global.currentSelectLevel-1][0];this.gameStarOne=Global.gameTargetData[Global.currentSelectLevel-1][1];this.gameStarTwo=Global.gameTargetData[Global.currentSelectLevel-1][2];this.gameStarThree=Global.gameTargetData[Global.currentSelectLevel-1][3];this.gameTargetScore=Global.gameTargetData[Global.currentSelectLevel-1][4];this.gameTargetOneID=Global.gameTargetData[Global.currentSelectLevel-1][5];this.gameTargetOneNum=Global.gameTargetData[Global.currentSelectLevel-1][6];this.gameTargetTwoID=Global.gameTargetData[Global.currentSelectLevel-1][7];this.gameTargetTwoNum=Global.gameTargetData[Global.currentSelectLevel-1][8];this.gameInitFallCD=Global.gameTargetData[Global.currentSelectLevel-1][9];this.gameFallCD=Global.gameTargetData[Global.currentSelectLevel-1][10];this.calculateFallToothTime();this.isCanAddTimeMonster=false;this.addTimeMonsterID=0;},initData:function(){this.tempMonsterArrs=[];this.removeRows=[];this.connectMonsterArrs=[];this.tempColsArr=[0,0,0,0,0,0,0];this.tileMap=new TileMap();var tileMapData=this.createTileMapData(Global.classicModeOffsetX,Global.classicModeOffsetY,Global.tileMapWidth,Global.tileMapHeight,Global.classicModeWidth,Global.classicModeHeight,null);this.tileMap.initialize(tileMapData);this.createMapPoint();this.initMap(0);},createMapPoint:function(){this.tileMapPos=[];for(var row=0;row<Global.classicModeHeight;row++){var arr=[];for(var col=0;col<Global.classicModeWidth;col++){var point=cc.p(col\*Global.tileMapWidth+Global.monsterWidth/2+43,Global.tileMapHeight\*(row-3)+Global.monsterHeight/2);arr.push(point);}

this.tileMapPos.push(arr);}},createTileMapData:function(offsetX,offsetY,gridWidth,gridHeight,gridMaxX,gridMaxY,defaultGridValue){var data=new Object();data.offsetX=offsetX;data.offsetY=offsetY;data.gridWidth=gridWidth;data.gridHeight=gridHeight;data.gridMaxX=gridMaxX;data.gridMaxY=gridMaxY;data.defaultGridValue=defaultGridValue;return data;},initUI:function(){this.rootNode=loadCSS(res.LevelGameLayer\_json);this.addChild(this.rootNode);this.btnPause=this.rootNode.Button\_1;this.btnAddGold=this.rootNode.Image\_8.Button\_2;this.btnAddGold.setPressedActionEnabled(true);this.btnDyeBottle=this.rootNode.Image\_9;this.rootNode.dyebottlex\_1.visible=false;this.btnToiletSucker=this.rootNode.Image\_12;this.rootNode.toiletsuckerx\_2.visible=false;this.rootNode.Image\_1.setLocalZOrder(-3);this.rootNode.GameTongue\_2.setLocalZOrder(-2);this.touchPanel=new cc.Layer();this.touchPanel.width=720;this.touchPanel.height=960;this.touchPanel.x=3;this.touchPanel.y=180;this.rootNode.addChild(this.touchPanel,-1);this.gameControl=currentScene.gameControl;this.rootNode.GameAlarm\_1.visible=false;this.initGameTargetUI();this.refreshCurrentMoney();this.refreshBtnAddMoney();},initGameTargetUI:function(){if(this.gameTargetScore>0){this.gameTarget.push(11);this.gameTarget.push(this.gameTargetScore);this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[10]);this.rootNode.BitmapFontLabel\_5.setString(this.gameTargetScore+"");if(this.gameTargetOneID>0){this.gameTarget.push(this.gameTargetOneID);this.gameTarget.push(this.gameTargetOneNum);this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID-1]);this.rootNode.BitmapFontLabel\_5\_0.setString("0/"+this.gameTargetOneNum);}else{if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.BitmapFontLabel\_5\_0.setString("0/"+this.gameTargetTwoNum);}else{this.rootNode.GameTargetGreen\_29.visible=false;this.rootNode.BitmapFontLabel\_5\_0.visible=false;this.rootNode.GameTargetBotton\_27.visible=false;this.rootNode.GameTarget\_24.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBlue\_28.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBotton\_26.x+=levelGameLayerTargetOffset;this.rootNode.BitmapFontLabel\_5.x+=levelGameLayerTargetOffset;}}}else{if(this.gameTargetOneID>0){this.gameTarget.push(this.gameTargetOneID);this.gameTarget.push(this.gameTargetOneNum);this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID-1]);this.rootNode.BitmapFontLabel\_5.setString("0/"+this.gameTargetOneNum);if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.GameTargetGreen\_29.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.BitmapFontLabel\_5\_0.setString("0/"+this.gameTargetTwoNum);}else{this.rootNode.GameTargetGreen\_29.visible=false;this.rootNode.BitmapFontLabel\_5\_0.visible=false;this.rootNode.GameTargetBotton\_27.visible=false;this.rootNode.GameTarget\_24.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBlue\_28.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBotton\_26.x+=levelGameLayerTargetOffset;this.rootNode.BitmapFontLabel\_5.x+=levelGameLayerTargetOffset;}}else{if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.GameTargetBlue\_28.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.BitmapFontLabel\_5.setString("0/"+this.gameTargetTwoNum);this.rootNode.GameTargetGreen\_29.visible=false;this.rootNode.BitmapFontLabel\_5\_0.visible=false;this.rootNode.GameTargetBotton\_27.visible=false;this.rootNode.GameTarget\_24.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBlue\_28.x+=levelGameLayerTargetOffset;this.rootNode.GameTargetBotton\_26.x+=levelGameLayerTargetOffset;this.rootNode.BitmapFontLabel\_5.x+=levelGameLayerTargetOffset;}else{this.rootNode.GameTargetGreen\_29.visible=false;this.rootNode.BitmapFontLabel\_5\_0.visible=false;this.rootNode.GameTargetBotton\_27.visible=false;}}}},refreshBtnAddMoney:function(){if(gamePlayerData.isJiHuo==0){this.btnAddGold.visible=false;}

if(gamePlayerData.isJiHuo==1){this.btnAddGold.visible=true;}},refreshTargetText:function(){if(this.gameTarget[0]==11){cc.log("第一个目标是分数");if(this.gameTarget[2]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[2]-1];var str=tempnum+"/"+this.gameTarget[3];this.rootNode.BitmapFontLabel\_5\_0.setString(str);}}else{if(this.gameTarget[0]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[0]-1];var str=tempnum+"/"+this.gameTarget[1];this.rootNode.BitmapFontLabel\_5.setString(str);if(this.gameTarget[2]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[2]-1];var str=tempnum+"/"+this.gameTarget[3];this.rootNode.BitmapFontLabel\_5\_0.setString(str);}}}},refreshCurrentScore:function(recordConnectRemoveTimes,removeCount){var addScore=(recordConnectRemoveTimes\*100+300)\*(removeCount-2);this.refreshCurrentScoreByScore(addScore);currentScene.createEncourage(addScore);},refreshCurrentScoreByScore:function(score){if(!this.isGameWin&&!this.isGameOver){this.score+=score;this.rootNode.Image\_4.BitmapFontLabel\_2.setString(this.score+"");}},initText:function(){this.rootNode.BitmapFontLabel\_1.setString(""+Global.currentSelectLevel);this.rootNode.Image\_4.BitmapFontLabel\_2.setString(""+this.score);this.rootNode.Image\_8.BitmapFontLabel\_3.setString(""+gamePlayerData.money);},refreshCurrentMoney:function(){this.rootNode.Image\_8.BitmapFontLabel\_3.setString(""+gamePlayerData.money);},initListener:function(){this.btnPause.addTouchEventListener(this.btnPauseFun,this.btnPause);this.btnAddGold.addTouchEventListener(this.btnAddGoldFun,this.btnAddGold);engine.gameEvent.createMouseDown(this.btnDyeBottle,this.btnDyeBottleFun);engine.gameEvent.createMouseDown(this.btnToiletSucker,this.btnToiletSuckerFun);this.addTouchListener();},btnDyeBottleFun:function(){playEffect(2);cc.log("btnDyeBottleFun");if(currentScene.gameLayer.isSelectToiletSucker)return;if(currentScene.gameLayer.isSelectDyeBottle){if(currentScene.gameLayer.isDyeing){cc.log("正在染色");}else{currentScene.gameLayer.isSelectDyeBottle=false;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=false;}}else{if(gamePlayerData.money>=500){currentScene.gameLayer.isSelectDyeBottle=true;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=true;}}

savePlayerData();},btnToiletSuckerFun:function(){playEffect(2);cc.log("btnToiletSuckerFun");if(currentScene.gameLayer.isSelectDyeBottle)return;if(currentScene.gameLayer.isSelectToiletSucker){if(currentScene.gameLayer.isPlayToilet){cc.log("正在播放马桶塞动画");}else{currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;}}else{if(gamePlayerData.money>=200){currentScene.gameLayer.isSelectToiletSucker=true;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=true;}}

savePlayerData();},btnPauseFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(4);cc.log("btnPauseFun");if(!this.isGameOver&&!this.isGameWin){this.isPause=true;cc.director.pause();currentScene.gamePause();}

break;}},btnAddGoldFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnAddGoldFun");createSDKLayerFun();break;}},initMap:function(row){for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var gameMap=Global.gameMaps[Global.currentSelectLevel-1];var monsterType=gameMap[(9-(i-3))-1][j];this.createMonster(i,j,monsterType);}}},createMonster:function(i,j,monsterType){if(monsterType<=0)return;var monster=new SpecialMonsterNode(monsterType,i,j);if(monster!=null){monster.x=this.tileMapPos[i][j].x;monster.y=this.tileMapPos[i][j].y;this.touchPanel.addChild(monster,Global.classicModeHeight-(i-3));monster.rowcol=[i,j];monster.row=monster.rowcol[0];this.tileMap.setGridInfo(i,j,monster);}},fallMonstersCallFun:function(){this.searchConnectedMonsters(this.tileMap);var isConnected=this.checkIsConnectedMonsters();if(isConnected){if(this.touchMonster==null){if(!this.isTouchDown){var isFallDowaning=this.searchIsFallDown(this.tileMap);var isMonsterShake=this.searchIsShakeMonster(this.tileMap);if(isFallDowaning&&isMonsterShake){this.upInitMonster();}}}}else{this.searchConnectedMonstersArrs();this.shakeConnectedMonsters();}},checkIsConnectedMonsters:function(){if(this.tempMonsterArrs==null)return;var connectedMonstersTimes=0;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){connectedMonstersTimes++;}}}}

if(connectedMonstersTimes==0){return true;}else{return false;}},initTileMapDown:function(row){this.createMonsterByRows(row,2);},createMonsterByRows:function(rows,upAndDown){var monsterTypeArr=[];for(var j=0;j<rows;j++){for(var i=0;i<7;i++){var monsterType=randomNums(1,8);if(monsterType==6){monsterType=randomNums(1,6);}

if(!this.isCanAddTimeMonster){if(monsterType==7){monsterType=randomNums(1,6);}}else{if(monsterType==7){this.isCanAddTimeMonster=false;}}

if(i>=2){if(monsterTypeArr[i-1]==monsterType&&monsterTypeArr[i-2]==monsterType){if(monsterType!=1){monsterType=monsterType-1;}else{monsterType=2;}}}

monsterTypeArr.push(monsterType);if(upAndDown==1){if(this.tileMap.gridArr[Global.classicModeHeight-1][i]==null&&this.tileMap.gridArr[Global.classicModeHeight-3][i]==null){var monster=new SpecialMonsterNode(monsterType,Global.classicModeHeight-1,i);monster.x=this.tileMapPos[Global.classicModeHeight-1][i].x;monster.y=this.tileMapPos[Global.classicModeHeight-1][i].y;this.touchPanel.addChild(monster);this.tileMap.setGridInfo(Global.classicModeHeight-1,i,monster);}}

if(upAndDown==2){if(this.tileMap.gridArr[3-1-j+1][i]==null){var monster=new SpecialMonsterNode(monsterType,3-1-j+1,i);monster.x=this.tileMapPos[3-1-j][i].x;monster.y=this.tileMapPos[3-1-j][i].y;this.touchPanel.addChild(monster);this.tileMap.setGridInfo(3-1-j,i,monster);this.gameControl.upMonster(3-1-j,i,monster,rows);}}}}},monsterCallFun:function(target,data){if(data==null)return;var monster=data[0];var j=data[1];var i=data[2];monster.rowcol=[j,i];monster.setLocalZOrder(Global.classicModeHeight-(j-3));monster.row=monster.rowcol[0];this.tileMap.setGridInfo(j,i,monster);},searchRowsOfTileMap:function(tileMap,isFull){if(tileMap==null)return;var tempNum=0;var num=0;if(isFull){for(var i=3;i<tileMap.gridMaxY-1;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){num++;}

if(j==tileMap.gridMaxX-1){if(num==tileMap.gridMaxX){tempNum++;}

num=0;}}}}else{for(var \_i=3;\_i<tileMap.gridMaxY-1;\_i++){for(var \_j=0;\_j<tileMap.gridMaxX;\_j++){if(tileMap.gridArr[\_i][\_j]!=null){tempNum++;break;}}}}

return tempNum;},searchColsOfTileMap:function(tileMap){if(tileMap==null)return;var tempNum=[];for(var i=0;i<tileMap.gridMaxX;i++){if(tileMap.gridArr[tileMap.gridMaxY-1-1][i]!=null){tempNum.push(i);}}

return tempNum;},searchConnectedMonsters:function(tileMap){this.tempMonsterArrs=[];var arr=[];for(var i=3;i<tileMap.gridMaxY-2;i++){for(var j=0;j<tileMap.gridMaxX;j++){this.tempMonsterArr=[];if(tileMap.gridArr[i][j]!=null){if(tileMap.gridArr[i][j].isMonster||tileMap.gridArr[i][j].isTimeMonster||tileMap.gridArr[i][j].isCageMonster||tileMap.gridArr[i][j].isBoxMonster||tileMap.gridArr[i][j].isAdvancedBoxMonster||tileMap.gridArr[i][j].isMonsterKing){if(!tileMap.gridArr[i][j].isSearched&&!tileMap.gridArr[i][j].isWaiteDie&&!tileMap.gridArr[i][j].isDown&&!tileMap.gridArr[i][j].isUp){if(i==3||tileMap.gridArr[i-1][j]!=null||tileMap.gridArr[i][j].isBoxMonster||tileMap.gridArr[i][j].isAdvancedBoxMonster){tileMap.gridArr[i][j].isSearched=true;this.tempMonsterArr.push(tileMap.gridArr[i][j]);this.searchMonstersByDir(i,j,tileMap,0,1);this.searchMonstersByDir(i,j,tileMap,0,-1);this.searchMonstersByDir(i,j,tileMap,-1,0);this.searchMonstersByDir(i,j,tileMap,1,0);this.tempMonsterArrs.push(this.tempMonsterArr);}}}}}}

for(var i=3;i<tileMap.gridMaxY-2;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){tileMap.gridArr[i][j].isSearched=false;}}}},searchMonstersByDir:function(i,j,tileMap,horizontal,vertical){if(i+vertical>2&&i+vertical<13){if(tileMap.isExistGrid(j+horizontal,i+vertical)){if(tileMap.gridArr[i+vertical][j+horizontal]!=null){if(tileMap.gridArr[i+vertical][j+horizontal].isMonster||tileMap.gridArr[i+vertical][j+horizontal].isTimeMonster||tileMap.gridArr[i+vertical][j+horizontal].isCageMonster||tileMap.gridArr[i+vertical][j+horizontal].isBoxMonster||tileMap.gridArr[i+vertical][j+horizontal].isAdvancedBoxMonster||tileMap.gridArr[i+vertical][j+horizontal].isMonsterKing){if((tileMap.gridArr[i+vertical][j+horizontal].skin==tileMap.gridArr[i][j].skin)){if(!tileMap.gridArr[i+vertical][j+horizontal].isSearched&&!tileMap.gridArr[i+vertical][j+horizontal].isWaiteDie&&!tileMap.gridArr[i+vertical][j+horizontal].isDown&&!tileMap.gridArr[i+vertical][j+horizontal].isUp){if((i+vertical==3)||tileMap.gridArr[i+vertical-1][j+horizontal]!=null||tileMap.gridArr[i+vertical][j+horizontal].isBoxMonster||tileMap.gridArr[i+vertical][j+horizontal].isAdvancedBoxMonster){tileMap.gridArr[i+vertical][j+horizontal].isSearched=true;this.tempMonsterArr.push(tileMap.gridArr[i+vertical][j+horizontal]);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,0,1);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,0,-1);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,-1,0);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,1,0);}}}}}}}},searchConnectedMonstersArrs:function(){if(this.tempMonsterArrs==null)return;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){this.connectMonsterArrs.push(this.tempMonsterArrs[i]);}}}}},shakeConnectedMonsters:function(){if(this.connectMonsterArrs==null)return;var connectMonsterArrsLength=this.connectMonsterArrs.length;for(var i=0;i<connectMonsterArrsLength;i++){var plength=this.connectMonsterArrs[i].length;if(plength>=3){var skin=this.connectMonsterArrs[i][0].skin;var isExistArr=false;var shakeTimes=Global.monsterShakeTimes;for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(this.connectMonsterArrs[i][j].isShake){var removeAct=this.gameControl.searchExistArrFromRemoveList(this.connectMonsterArrs[i][j]);isExistArr=true;break;}}

if(isExistArr){for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(!this.connectMonsterArrs[i][j].isShake){if(removeAct!=null){if(removeAct.arr!=null){this.connectMonsterArrs[i][j].isShake=true;playEffect(19);this.connectMonsterArrs[i][j].playShakeAni();removeAct.times=removeAct.times+5;removeAct.arr.push(this.connectMonsterArrs[i][j]);playEffect(20);}}}}}else{var isMonsterRemove=false;for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(this.connectMonsterArrs[i][j].isRemve){this.connectMonsterArrs[i][j].isRemve=false;isMonsterRemove=true;}}

for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(!this.connectMonsterArrs[i][j].isShake&&!this.connectMonsterArrs[i][j].isEnd&&!this.connectMonsterArrs[i][j].isRemve){this.connectMonsterArrs[i][j].isShake=true;this.connectMonsterArrs[i][j].playShakeAni();if(j==plength-1){playEffect(19);}}}

this.gameControl.createRemoveList(this.connectMonsterArrs[i],shakeTimes,skin);}}}

this.connectMonsterArrs=[];},removeConnectedMonsters:function(){if(this.tempMonsterArrs==null)return;this.tempColsArr=[0,0,0,0,0,0,0];this.isClearMonster=true;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){}}}}

this.tempMonsterArr=[];this.tempMonsterArrs=[];},fallMonstersBySpace:function(){Global.monsterDown=false;if(this.isBanFall)return;this.setFallRemoveRows();for(var i=0;i<this.removeRows.length;i++){var row=this.removeRows[i];var space=0;for(var j=3;j<Global.classicModeHeight;j++){var monster=this.tileMap.gridArr[j][row];if(monster==null||monster.isBox||monster.isAdvancedBox){space++;}else{if(space>0){if(!monster.isShake&&!monster.isDown&&!monster.isWaiteDie){if(monster.isMonster||monster.isIce||monster.isTimeMonster||monster.isCageMonster||monster.isMonsterKing){this.gameControl.moveMonster(j,row,monster,1);}}

if(monster.isShake||monster.isBoxMonster||monster.isAdvancedBoxMonster||monster.isWaiteDie){space=0;}}}}}

this.removeRows=[];if(!Global.monsterDown){if(this.gameControl.isMoveEnd){if(currentScene.gameLayer.isTouchOnMouseUp){currentScene.gameLayer.isTouchOnMouseUp=false;}

var self=this;var callAction=cc.callFunc(function(){self.fallMonstersCallFun();});this.runAction(cc.sequence(cc.delayTime(0.2),callAction));}else{if(currentScene.gameLayer.isTouchOnMouseUp){currentScene.gameLayer.isTouchOnMouseUp=false;var self=this;var callAction=cc.callFunc(function(){self.fallMonstersCallFun();});this.runAction(cc.sequence(cc.delayTime(0.2),callAction));}}}},fallMonster:function(){if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver)return;this.createMonsterByRows(1,1);this.fallMonstersBySpace();},upInitMonster:function(){if(currentScene.gameLayer.isTouchDown||currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver)return;if(currentScene.gameControl.moveList.length!=0||currentScene.gameControl.removeList.length!=0||currentScene.gameControl.removeSpecialMonsterList.length!=0||currentScene.gameControl.upMonsterList.length>1||currentScene.gameControl.downMonsterList.length!=0){if(currentScene.gameControl.removeSpecialMonsterList.length!=0||currentScene.gameControl.removeList.length!=0){return;}else{return;}}

if(!this.isCreateMonsterDown){this.printArr();this.searchConnectedMonsters(this.tileMap);var isConnected=this.checkIsConnectedMonsters();if(!isConnected){this.searchConnectedMonstersArrs();this.shakeConnectedMonsters();}else{var MonsterRowsCols=this.computeCreateMonsterDownRows(this.tileMap);var rows=MonsterRowsCols[0];if(rows==0)return;this.isCreateMonsterDown=true;var cols=MonsterRowsCols[1];var tileMapRows=this.searchRowsOfTileMap(this.tileMap,false);for(var i=tileMapRows+3;i>=3;i--){for(var j=0;j<Global.classicModeWidth;j++){var boxCow=-1;for(var m=3;m<Global.classicModeHeight-2;m++){if(this.tileMap.gridArr[m][j]!=null){if(this.tileMap.gridArr[m][j].isAdvancedBox||this.tileMap.gridArr[m][j].isBox||this.tileMap.gridArr[m][j].isAdvancedBoxMonster||this.tileMap.gridArr[m][j].isBoxMonster){boxCow=m;break;}}}

if(boxCow==-1||i<boxCow){if(i<Global.classicModeHeight-2&&i+1<=Global.classicModeHeight-3){if(this.tileMap.gridArr[i+1][j]==null){var tileMapMonster=this.tileMap.gridArr[i][j];if(tileMapMonster!=null&&!tileMapMonster.isBox&&!tileMapMonster.isBoxMonster&&!tileMapMonster.isAdvancedBox&&!tileMapMonster.isAdvancedBoxMonster){if(!tileMapMonster.isDown&&!tileMapMonster.isShake&&!tileMapMonster.isWaiteDie&&!tileMapMonster.isUp){this.gameControl.upMonster(i,j,tileMapMonster,rows);}}}}}}}

this.createMonsterByRows(rows,2);}}},getPosByGrid:function(Grid){return this.tileMapPos[Grid[1]][Grid[0]];},addTouchListener:function(){engine.gameEvent.createMouseDown(this.touchPanel,this.onMouseDown);engine.gameEvent.createMouseMove(this.touchPanel,this.onMouseMove);engine.gameEvent.createMouseUp(this.touchPanel,this.onMouseUp);},onMouseDown:function(touch,event){currentScene.gameLayer.isTouchDown=true;if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver||currentScene.gameLayer.isBanFall)return;var pos=touch.getLocation();if(currentScene.gameLayer.touchMonster==null){currentScene.gameLayer.selectTouchMonster(pos);}

return true;},selectTouchMonster:function(pos){if(!currentScene.gameLayer.isCanTouchDown)return;currentScene.gameLayer.isCanTouchDown=false;currentScene.gameLayer.touchPosition=pos;if(!currentScene.gameLayer.checkTouchArea(pos))return;var tilePos=currentScene.gameLayer.getTileMapPosByPoint(pos);var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,tilePos);grid[1]=grid[1]+3;if((currentScene.gameLayer.isSelectDyeBottle||currentScene.gameLayer.isSelectToiletSucker)&&!currentScene.gameLayer.isDyeing&&!currentScene.gameLayer.isPlayToilet){if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]==null&&currentScene.gameLayer.isSelectToiletSucker){currentScene.gameLayer.isPlayToilet=false;currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;}else{var props=new PropsNode();if(currentScene.gameLayer.isSelectDyeBottle){props.type=1;currentScene.gameLayer.isDyeing=true;props.initProps(res.dyeingbottle\_json,"dyeingbottle",currentScene.gameLayer,grid);}

if(currentScene.gameLayer.isSelectToiletSucker){props.type=2;currentScene.gameLayer.isPlayToilet=true;props.initProps(res.toiletplug\_json,"toiletplug",currentScene.gameLayer,grid);}

var tempPos=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]];props.setPosition(tempPos);props.x=props.x+3;props.y=props.y+180;currentScene.gameLayer.addChild(props,50);}}else{currentScene.gameLayer.frontGrid=grid;if(!currentScene.gameLayer.checkMonsterMove(grid)){currentScene.gameLayer.isTouchDown=true;if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isCreateBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){var monster=currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]],1,tilePos);currentScene.gameLayer.touchMonster=monster;}else if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isCreateAdvanceBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){var monster=currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]],2,tilePos);currentScene.gameLayer.touchMonster=monster;cc.log("grid[1] :"+grid[1]+" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "+"grid[0] :"+grid[0]);}else{if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=true;currentScene.gameLayer.touchMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];}}

if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;return;}

if(currentScene.gameLayer.touchMonster.isMonster||currentScene.gameLayer.touchMonster.isTimeMonster||currentScene.gameLayer.touchMonster.isMonsterKing){if(currentScene.gameLayer.touchMonster.isFalling){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;return;}

if(currentScene.gameLayer.touchMonster.isShake)currentScene.gameLayer.touchMonster=null;if(currentScene.gameLayer.touchMonster!=null){currentScene.gameLayer.touchMonster.isDown=false;currentScene.gameLayer.touchMonster.isMove=true;if(!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]=null;}

if(currentScene.gameLayer.removeRows.indexOf(grid[0])==-1){currentScene.gameLayer.removeRows.push(grid[0]);}}}}}},onMouseMove:function(touch,event){if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver||currentScene.gameLayer.isBanFall)return;var pos=touch.getLocation();if(!currentScene.gameLayer.checkTouchArea(pos))return;if(currentScene.gameLayer.touchMonster==null){currentScene.gameLayer.selectTouchMonster(pos);}

var delta=touch.getDelta();var tilePos=currentScene.gameLayer.getTileMapPosByPoint(pos);if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster)return;var currentPoint=currentScene.gameLayer.touchMonster.getPosition();var deltaX=tilePos.x-currentPoint.x;var deltaY=tilePos.y-currentPoint.y;var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,tilePos);grid[1]=grid[1]+3;currentScene.gameLayer.currentGrid=grid;if(currentScene.gameLayer.touchPosition!=null){if((currentScene.gameLayer.currentGrid[0]!=currentScene.gameLayer.frontGrid[0]||currentScene.gameLayer.currentGrid[1]!=currentScene.gameLayer.frontGrid[1])&&(Math.abs(tilePos.x-currentScene.gameLayer.touchPosition.x)>Global.tileMapWidth)){currentScene.gameLayer.touchPosition=null;currentScene.gameLayer.fallMonstersBySpace();}}

var monster=null;if(grid[0]>=0&&grid[0]<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-2){monster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];}

var dirMonster=null;if(currentScene.gameLayer.checkOutScreen(tilePos)){if(currentScene.gameLayer.checkMonsterMove(grid))return;if(monster!=null){currentScene.gameLayer.isTouchInBox=true;var tilePosDeltaX=tilePos.x-currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;if(Math.abs(tilePosDeltaX)<=15){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-2||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(grid[0]+1>=Global.classicModeWidth||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]-1][grid[0]];if(grid[1]-1>2){if(dirMonster!=null){}else{tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}

if(tilePosDeltaX<0&&Math.abs(tilePosDeltaX)>15&&Math.abs(tilePosDeltaX)<45){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-2||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(grid[0]+1>=Global.classicModeWidth||dirMonster!=null){cc.log("上左右 都已满");dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]-1][grid[0]];if(grid[1]-1>2){if(dirMonster!=null){cc.log("上左右下 都已满");}else{tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}

if(tilePosDeltaX>0&&Math.abs(tilePosDeltaX)>15&&Math.abs(tilePosDeltaX)<45){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(grid[0]+1>=Global.classicModeWidth||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-2||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){cc.log("上左右 都已满");dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]-1][grid[0]];if(grid[1]-1>2){if(dirMonster!=null){cc.log("上左右下 都已满");}else{tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]-1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}

currentScene.gameLayer.touchMonster.setPosition(tilePos);}else{if(currentScene.gameLayer.isTouchInBox){currentScene.gameLayer.isTouchInBox=false;if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].y;}}else{if(deltaX>0){if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,1,0)){tilePos.x=currentPoint.x;}}else{if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,-1,0)){tilePos.x=currentPoint.x;}}

if(deltaY>0){if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,0,1)){tilePos.y=currentPoint.y;}}else{if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,0,-1)){tilePos.y=currentPoint.y;}}}

currentScene.gameLayer.touchMonster.setPosition(tilePos);}}},onMouseUp:function(touch,event){if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster){currentScene.gameLayer.touchMonster=null;currentScene.gameLayer.currentGrid=null;currentScene.gameLayer.frontGrid=null;return;}

var currentPoint=currentScene.gameLayer.touchMonster.getPosition();var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,currentPoint);grid[1]=grid[1]+3;var gridPos=currentScene.gameLayer.getPosByGrid(grid);if(currentScene.gameLayer.touchMonster!=null){currentScene.gameLayer.isTouchDown=false;currentScene.gameLayer.touchMonster.isMove=false;currentScene.gameLayer.touchMonster.setPosition(gridPos);if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox){var skin=currentScene.gameLayer.touchMonster.skin;currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin+1);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin=skin;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster=true;}else if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox){var skin=currentScene.gameLayer.touchMonster.skin;currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin+1);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin=skin;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster=true;cc.log("grid[1] :"+grid[1]+" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "+"grid[0] :"+grid[0]);}else{if(grid[0]-1>=0&&grid[0]-1<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-2&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].rowcol=[grid[1],grid[0]-1];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{if(grid[0]+1>=0&&grid[0]+1<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-2&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].rowcol=[grid[1],grid[0]+1];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{if(grid[0]>=0&&grid[0]<Global.classicModeWidth&&grid[1]+1>=3&&grid[1]+1<Global.classicModeHeight-2&&currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]);currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].rowcol=[grid[1]+1,grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].setLocalZOrder(Global.classicModeHeight-(grid[1]+1-3));}else{var isHaveEmptyGrid=false;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){if(currentScene.gameLayer.tileMap.gridArr[i][j]==null&&!isHaveEmptyGrid){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[i][j]);currentScene.gameLayer.tileMap.gridArr[i][j]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[i][j].isDelete=false;currentScene.gameLayer.tileMap.gridArr[i][j].rowcol=[i,j];currentScene.gameLayer.tileMap.gridArr[i][j].row=currentScene.gameLayer.tileMap.gridArr[i][j].rowcol[0];currentScene.gameLayer.tileMap.gridArr[i][j].setLocalZOrder(Global.classicModeHeight-(i+1-3));isHaveEmptyGrid=true;}}}

if(!isHaveEmptyGrid){currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.touchMonster=null;}}}}}}}

currentScene.gameLayer.isTouchOnMouseUp=true;currentScene.gameLayer.fallMonstersBySpace();currentScene.gameLayer.touchMonster=null;currentScene.gameLayer.currentGrid=null;currentScene.gameLayer.frontGrid=null;currentScene.gameLayer.isTouchDown=false;},setFallRemoveRows:function(){this.removeRows=[];for(var i=0;i<Global.classicModeWidth;i++){this.removeRows.push(i);}},checkTouchArea:function(pos){if(pos.x<718&&pos.x>2&&pos.y<960&&pos.y>180){return true;}else{return false;}},getTileMapGirdByPoint:function(tileMap,pos){var posX=pos.x-43;var posY=pos.y;var lx=parseInt(posX/tileMap.gridWidth);var ly=parseInt(posY/tileMap.gridHeight);return[lx,ly];},getTileMapPosByPoint:function(pos){return cc.p(pos.x-this.touchPanel.x,pos.y-this.touchPanel.y);},checkOutScreen:function(pos){if(pos.x-Global.monsterWidth/2>25&&pos.y-Global.monsterHeight/2>5&&pos.x+Global.monsterWidth/2<Global.classicModeWidth\*Global.tileMapWidth+50&&pos.y+Global.monsterHeight/2<Global.classicModeHeight\*Global.tileMapHeight+5){return true;}else{return false;}},checkMonsterByDirPos:function(pos,horizontal,vertical){var lx=parseInt((pos.x+horizontal-43+horizontal\*this.tileMap.gridWidth/2)/this.tileMap.gridWidth);var ly=parseInt((pos.y+vertical+vertical\*this.tileMap.gridHeight/2)/this.tileMap.gridHeight);ly=ly+3;if(this.tileMap.gridArr[ly][lx]==null){return false;}else{return true;}},checkMonsterMove:function(gird){var bool1=this.checkGrid(gird[0],gird[1]-1);var bool2=this.checkGrid(gird[0]-1,gird[1]);var bool3=this.checkGrid(gird[0]+1,gird[1]);var bool4=this.checkGrid(gird[0],gird[1]+1);var bool=bool1+bool2+bool3+bool4;if(bool==4){return true;}else{return false;}},checkGrid:function(gird0,gird1){if(gird0>=0&&gird0<Global.classicModeWidth&&gird1>=3&&gird1<Global.classicModeHeight-2){if(this.tileMap.gridArr[gird1][gird0]!=null){return 1;}else{return 0;}}else{return 1;}},computeCreateMonsterDownRows:function(tileMap){if(tileMap==null)return;var allMonsterNum=0;var rows=0;for(var i=3;i<tileMap.gridMaxY-1;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){allMonsterNum++;}}}

if(allMonsterNum>32){rows=0;}else{if(allMonsterNum>=21&&allMonsterNum<28){rows=0;}

if(allMonsterNum>=14&&allMonsterNum<21){rows=1;}

if(allMonsterNum>=7&&allMonsterNum<14){rows=1;}

if(allMonsterNum<7){rows=1;}}

var cols=this.searchColsOfTileMap(tileMap);return[rows,cols];},searchIsFallDown:function(tileMap){var times=0;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=tileMap.gridArr[i][j];if(monster!=null){if(monster.isFalling){times++;}}}}

if(times==0){return true;}else{return false;}},searchIsShakeMonster:function(tileMap){var times=0;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=tileMap.gridArr[i][j];if(monster!=null){if(monster.isShake){times++;}}}}

if(times==0){return true;}else{return false;}},printArr:function(){for(var i=Global.classicModeHeight-3;i>2;i--){var temp="";for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null){switch(monster.skin){case 0:temp=temp+""+"蓝";break;case 1:temp=temp+""+"绿";break;case 2:temp=temp+""+"紫";break;case 3:temp=temp+""+"红";break;case 4:temp=temp+""+"黄";break;}}else{temp=temp+""+0;}}}},starlightMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;for(var i=-1;i<2;i++){for(var j=-1;j<2;j++){var tempMonster=this.tileMap.gridArr[colRow[0]+i][colRow[1]+j];if(colRow[0]+i>=3&&colRow[0]+i<Global.classicModeHeight-2&&colRow[1]+j>=0&&colRow[1]+j<Global.classicModeWidth&&tempMonster!=null){if(i==0&&j==0){}else{if(!tempMonster.isShake&&!tempMonster.isWaiteDie&&!tempMonster.isDown&&!tempMonster.isUp){playEffect(22);tempMonster.initDieEffectsType(1);if(tempMonster.effectsType==1){tempMonster.flameMonsterDie();}else{tempMonster.playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(tempMonster);}}}}}},flameMonsterFall:function(monster){if(monster==null)return;var colRow=monster.rowcol;if(monster.row==3){monster.playDieAni();}else{if(this.tileMap.gridArr[monster.row-1][colRow[1]]!=null&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isShake&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isWaiteDie&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isDown&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isUp){playEffect(22);this.tileMap.gridArr[monster.row-1][colRow[1]].initDieEffectsType(6);if(this.tileMap.gridArr[monster.row-1][colRow[1]].effectsType==1){this.tileMap.gridArr[monster.row-1][colRow[1]].flameMonsterDie();}else{this.tileMap.gridArr[monster.row-1][colRow[1]].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[monster.row-1][colRow[1]]);}

var moveAction=cc.moveBy(0.03,cc.p(0,-90));var self=this;var callBackAction=cc.callFunc(function(){monster.row=monster.row-1;self.flameMonsterFall(monster);});monster.runAction(cc.sequence(cc.delayTime(0.05),moveAction,callBackAction));}},flameMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;var row=monster.row;if(row==3){cc.log("移到底炸周边六个");if(colRow[1]-1>=0&&colRow[1]-1<Global.classicModeWidth&&this.tileMap.gridArr[row][colRow[1]-1]!=null){if(!this.tileMap.gridArr[row][colRow[1]-1].isWaiteDie&&!this.tileMap.gridArr[row][colRow[1]-1].isShake&&!this.tileMap.gridArr[row][colRow[1]-1].isDown&&!this.tileMap.gridArr[row][colRow[1]-1].isUp){playEffect(22);this.tileMap.gridArr[row][colRow[1]-1].initDieEffectsType(6);if(this.tileMap.gridArr[row][colRow[1]-1].effectsType==1){this.tileMap.gridArr[row][colRow[1]-1].flameMonsterDie();}else{this.tileMap.gridArr[row][colRow[1]-1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1]-1]);}}

if(colRow[1]-1>=0&&colRow[1]-1<Global.classicModeWidth&&row+1>=3&&row+1<Global.classicModeHeight-2&&this.tileMap.gridArr[row+1][colRow[1]-1]!=null){if(!this.tileMap.gridArr[row+1][colRow[1]-1].isWaiteDie&&!this.tileMap.gridArr[row+1][colRow[1]-1].isShake&&!this.tileMap.gridArr[row+1][colRow[1]-1].isDown&&!this.tileMap.gridArr[row+1][colRow[1]-1].isUp){playEffect(22);this.tileMap.gridArr[row+1][colRow[1]-1].initDieEffectsType(6);if(this.tileMap.gridArr[row+1][colRow[1]-1].effectsType==1){this.tileMap.gridArr[row+1][colRow[1]-1].flameMonsterDie();}else{this.tileMap.gridArr[row+1][colRow[1]-1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row+1][colRow[1]-1]);}}

if(colRow[1]+1>=0&&colRow[1]+1<Global.classicModeWidth&&this.tileMap.gridArr[row][colRow[1]+1]!=null){if(!this.tileMap.gridArr[row][colRow[1]+1].isWaiteDie&&!this.tileMap.gridArr[row][colRow[1]+1].isShake&&!this.tileMap.gridArr[row][colRow[1]+1].isDown&&!this.tileMap.gridArr[row][colRow[1]+1].isUp){playEffect(22);this.tileMap.gridArr[row][colRow[1]+1].initDieEffectsType(6);if(this.tileMap.gridArr[row][colRow[1]+1].effectsType==1){this.tileMap.gridArr[row][colRow[1]+1].flameMonsterDie();}else{this.tileMap.gridArr[row][colRow[1]+1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1]+1]);}}

if(colRow[1]+1>=0&&colRow[1]+1<Global.classicModeWidth&&row+1>=3&&row+1<Global.classicModeHeight-2&&this.tileMap.gridArr[row+1][colRow[1]+1]!=null){if(!this.tileMap.gridArr[row+1][colRow[1]+1].isWaiteDie&&!this.tileMap.gridArr[row+1][colRow[1]+1].isShake&&!this.tileMap.gridArr[row+1][colRow[1]+1].isDown&&!this.tileMap.gridArr[row+1][colRow[1]+1].isUp){playEffect(22);this.tileMap.gridArr[row+1][colRow[1]+1].initDieEffectsType(6);if(this.tileMap.gridArr[row+1][colRow[1]+1].effectsType==1){this.tileMap.gridArr[row+1][colRow[1]+1].flameMonsterDie();}else{this.tileMap.gridArr[row+1][colRow[1]+1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row+1][colRow[1]+1]);}}}},lightningMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;for(var i=0;i<Global.classicModeWidth;i++){if(i!=colRow[1]&&this.tileMap.gridArr[colRow[0]][i]!=null){if(!this.tileMap.gridArr[colRow[0]][i].isWaiteDie&&!this.tileMap.gridArr[colRow[0]][i].isShake&&!this.tileMap.gridArr[colRow[0]][i].isDown&&!this.tileMap.gridArr[colRow[0]][i].isUp){playEffect(22);this.tileMap.gridArr[colRow[0]][i].initDieEffectsType(5);if(this.tileMap.gridArr[colRow[0]][i].effectsType==1){this.tileMap.gridArr[colRow[0]][i].flameMonsterDie();}else{this.tileMap.gridArr[colRow[0]][i].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[colRow[0]][i]);}}}

for(var j=0;j<9;j++){if(j+3!=colRow[0]&&this.tileMap.gridArr[j+3][colRow[1]]!=null){if(!this.tileMap.gridArr[j+3][colRow[1]].isWaiteDie&&!this.tileMap.gridArr[j+3][colRow[1]].isShake&&!this.tileMap.gridArr[j+3][colRow[1]].isDown&&!this.tileMap.gridArr[j+3][colRow[1]].isUp){playEffect(22);this.tileMap.gridArr[j+3][colRow[1]].initDieEffectsType(5);if(this.tileMap.gridArr[j+3][colRow[1]].effectsType==1){this.tileMap.gridArr[j+3][colRow[1]].flameMonsterDie();}else{this.tileMap.gridArr[j+3][colRow[1]].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[j+3][colRow[1]]);}}}},blackHoleMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;var self=this;var monsterArr=[];for(var j=colRow[0]-2;j<colRow[0]+3;j++){for(var i=colRow[1]-2;i<colRow[1]+3;i++){if(j>=3&&j<Global.classicModeHeight-2&&i>=0&&i<Global.classicModeWidth&&!(i==colRow[1]&&j==colRow[0])){var rotaMonster=this.tileMap.gridArr[j][i];if(rotaMonster!=null&&rotaMonster.isBox==false&&rotaMonster.isAdvancedBox==false&&rotaMonster.isBoxMonster==false&&rotaMonster.isAdvancedBoxMonster==false&&!rotaMonster.isWaiteDie&&!rotaMonster.isShake&&!rotaMonster.isDown&&!rotaMonster.isUp){cc.log("//播放吸入动画");rotaMonster.isWaiteDie=true;var pStart=rotaMonster.getPosition();var pEnd=monster.getPosition();rotaMonster.setMonsterAngle(pStart,pEnd);rotaMonster.addScheduleTime();rotaMonster.setMonsterData(pStart,pEnd);monsterArr.push(rotaMonster);}}}}

if(monsterArr.length!=0){this.gameControl.createRemoveCycloneMonsterList(monsterArr);}},useDyeingBottle:function(grid){var skin=randomNums(0,5);for(var i=-1;i<2;i++){for(var j=-1;j<2;j++){if(grid[0]+j>=0&&grid[0]+j<Global.classicModeWidth&&grid[1]+i>=3&&grid[1]+i<Global.classicModeHeight-2&&this.tileMap.gridArr[grid[1]+i][grid[0]+j]!=null&&(this.tileMap.gridArr[grid[1]+i][grid[0]+j].isCageMonster||this.tileMap.gridArr[grid[1]+i][grid[0]+j].isBoxMonster||this.tileMap.gridArr[grid[1]+i][grid[0]+j].isAdvancedBoxMonster||this.tileMap.gridArr[grid[1]+i][grid[0]+j].isMonster||this.tileMap.gridArr[grid[1]+i][grid[0]+j].isMonsterKing)){var monster=this.tileMap.gridArr[grid[1]+i][grid[0]+j];if(!monster.isShake&&!monster.isDown&&!monster.isUp){if(monster.isMonsterKing){monster.monsterKing.setSkin(Global.monsterKingSkin[skin]);monster.skin=skin;monster.setMonsterKingLv();}else{monster.monster.setSkin(Global.monsterSkin[skin]);monster.skin=skin;}}}}}

gamePlayerData.money=gamePlayerData.money-500;currentScene.gameLayer.refreshCurrentMoney();currentScene.gameLayer.isDyeing=false;currentScene.gameLayer.isSelectDyeBottle=false;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=false;this.fallMonstersCallFun();},useToiletPlug:function(grid){cc.log("useToiletPlug");if(this.tileMap.gridArr[grid[1]][grid[0]]!=null){var skin=this.tileMap.gridArr[grid[1]][grid[0]].skin;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null){if(monster.skin!=null&&monster.skin==skin){monster.initDieEffectsType(1);if(monster.effectsType==1){monster.flameMonsterDie();}else{monster.playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(monster);}}}}}

gamePlayerData.money=gamePlayerData.money-200;currentScene.gameLayer.refreshCurrentMoney();currentScene.gameLayer.isPlayToilet=false;currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;},removedFromBox:function(monster,type,pos){monster.monster.removeFromParent();monster.monster=null;var ID=monster.skin+1;monster.skin=-1;if(type==1){monster.isBoxMonster=false;monster.isBox=true;}

if(type==2){monster.isAdvancedBoxMonster=false;monster.isAdvancedBox=true;}

var tempMonster=new SpecialMonsterNode(ID);if(tempMonster!=null){tempMonster.x=pos.x;tempMonster.y=pos.y;this.touchPanel.addChild(tempMonster);}

return tempMonster;},updateCheckout:function(dt){if(!this.isGameOver&&!this.isGameWin){if(this.countTime>=0){if(this.checkIsFullMonster()){if(!this.isFullMonster){this.isFullMonster=true;this.checkGameLose();this.stopScheduleCheckout();}}else{this.checkFallTooth(this.countTime);this.checkShowGameAlarm();this.checkAddTimeMonster();if(this.score>=1000&&this.score<3000&&!this.isShowTipsTwo){this.showGameTipsByID(1);}

if(this.score>=3000&&!this.isShowTipsThree){this.showGameTipsByID(2);}

this.countTime--;this.refreshTime();this.checkGameWin();}}else{this.isTimeUP=true;this.checkGameLose();this.stopScheduleCheckout();}}},refreshTime:function(){var second=this.countTime%60;second=second>=10?second:"0"+second;var minute=parseInt(this.countTime/60);minute=minute>=10?minute:"0"+minute;var str=minute+":"+second;this.rootNode.BitmapFontLabel\_6.setString(str);},checkGameWin:function(){var firstTargetCondition=false;var secondTargetCondition=false;if(this.gameTarget[0]==11){if(this.score>=this.gameTargetScore){firstTargetCondition=true;if(!this.isPlayCompleteTargetOneEffect){this.isPlayCompleteTargetOneEffect=true;playEffect(27);}}

cc.log("第一个目标是分数");if(this.gameTarget[2]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[2]-1];if(tempnum>=this.gameTarget[3]){if(!this.isPlayCompleteTargetTwoEffect){this.isPlayCompleteTargetTwoEffect=true;playEffect(27);}

secondTargetCondition=true;}}else{secondTargetCondition=true;}}else{if(this.gameTarget[0]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[0]-1];if(tempnum>=this.gameTarget[1]){firstTargetCondition=true;if(!this.isPlayCompleteTargetOneEffect){this.isPlayCompleteTargetOneEffect=true;playEffect(27);}}

if(this.gameTarget[2]!=null){var tempnum=currentScene.gameRecordRemoveMonster[this.gameTarget[2]-1];var str=tempnum+"/"+this.gameTarget[3];this.rootNode.BitmapFontLabel\_5\_0.setString(str);if(tempnum>=this.gameTarget[3]){secondTargetCondition=true;if(!this.isPlayCompleteTargetTwoEffect){this.isPlayCompleteTargetTwoEffect=true;playEffect(27);}}}else{secondTargetCondition=true;}}else{secondTargetCondition=true;}}

if(firstTargetCondition&&secondTargetCondition){cc.log("弹胜利界面");this.isGameWin=true;var score=this.score;var starNum=this.calculateGetStar();if(gamePlayerData.levelStarData[Global.currentSelectLevel-1]!=null){if(starNum>=gamePlayerData.levelStarData[Global.currentSelectLevel-1]){gamePlayerData.levelStarData[Global.currentSelectLevel-1]=starNum;}}else{gamePlayerData.levelStarData[Global.currentSelectLevel-1]=starNum;}

if(Global.currentSelectLevel==gamePlayerData.currentLevel){if(gamePlayerData.currentLevel==Global.maxSelectLevel){Global.currentSelectLevel=gamePlayerData.currentLevel;}else{gamePlayerData.currentLevel++;Global.currentSelectLevel=gamePlayerData.currentLevel;}}else{if(Global.currentSelectLevel<gamePlayerData.currentLevel){Global.currentSelectLevel++;}}

currentScene.gameLayer.boomOneByOne();var callFunAction=cc.callFunc(function(){currentScene.gameLayer.showGameAlarm(false);currentScene.gameWin(score,starNum);});this.runAction(cc.sequence(cc.delayTime(1),callFunAction));}

savePlayerData();},checkGameLose:function(){cc.log("弹失败界面");this.isGameOver=true;var callFunAction=cc.callFunc(function(){currentScene.gameLayer.showGameAlarm(false);currentScene.gameLose();});var callFunActionI=cc.callFunc(function(){currentScene.gameLayer.playTimeUPAnimation();});if(currentScene.gameLayer.isTimeUP){currentScene.gameLayer.isTimeUP=false;this.runAction(cc.sequence(callFunActionI,cc.delayTime(1),callFunAction));}else{this.runAction(cc.sequence(cc.delayTime(1),callFunAction));}},calculateGetStar:function(){if(this.countTime>=this.gameStarThree){return 3;}else if(this.countTime>=this.gameStarTwo&&this.countTime<this.gameStarThree){return 2;}else if(this.countTime>=this.gameStarOne&&this.countTime<this.gameStarTwo){return 1;}else{cc.log("没有得星");return 0;}},showGameAlarm:function(boolean){if(boolean){this.rootNode.GameAlarm\_1.visible=true;this.rootNode.GameAlarm\_1.runAction(cc.sequence(cc.fadeIn(0.2),cc.fadeOut(0.5)).repeatForever());if(!this.isPlayWarningEffect){this.isPlayWarningEffect=true;playWarningEffect(25);}}else{this.rootNode.GameAlarm\_1.visible=false;if(this.isPlayWarningEffect){this.isPlayWarningEffect=false;stopWarningEffect();}}},checkShowGameAlarm:function(){var monsterCount=0;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null&&!monster.isShake){monsterCount++;}}}

if(this.countTime<=15||monsterCount>45){if(this.countTime==15){this.unschedule(this.fallMonster);this.schedule(this.fallMonster,4);}

if(this.countTime==10){this.unschedule(this.fallMonster);this.schedule(this.fallMonster,3);}

if(this.countTime==5){this.unschedule(this.fallMonster);this.schedule(this.fallMonster,2);}

if(!this.isShowGameAlarm){this.isShowGameAlarm=true;this.showGameAlarm(this.isShowGameAlarm);}}else{if(this.isShowGameAlarm){this.unschedule(this.fallMonster);this.schedule(this.fallMonster,10);}

this.isShowGameAlarm=false;this.showGameAlarm(this.isShowGameAlarm);}},addToothAnimation:function(type,pos){var sprite=new cc.Sprite("#tooth"+type+"0001"+".png");var frames=[];var picNum=0;if(type==1){picNum=16;}

if(type==2){picNum=19;}

for(var i=1;i<=picNum;i++){var frame=cc.spriteFrameCache.getSpriteFrame("tooth"+type+"000"+i+".png");frames.push(frame);}

var Animation=new cc.Animation.create(frames,0.04);var animate=new cc.Animate(Animation);var callFun=cc.callFunc(function(){sprite.removeFromParent();});sprite.runAction(cc.sequence(animate,callFun));sprite.setPosition(cc.p(pos.x-2,pos.y-10));this.addChild(sprite,5);},checkFallTooth:function(count){for(var i=0;i<12;i++){if(count==this.fallToothTime[i]){var ID=11-i;var tooth=this.rootNode.getChildByName("GameTooth"+ID);tooth.visible=false;var pos=tooth.getPosition();var type=2;if(ID==2||ID==5||ID==8){type=1;}

this.addToothAnimation(type,pos);}}},calculateFallToothTime:function(){this.fallToothTime=[];var timeOne=parseInt((this.countTime-this.gameStarThree)/4);this.fallToothTime.push(this.countTime-timeOne);this.fallToothTime.push(this.countTime-2\*timeOne);this.fallToothTime.push(this.countTime-3\*timeOne);this.fallToothTime.push(this.gameStarThree);var timeTwo=parseInt((this.gameStarThree-this.gameStarTwo)/3);this.fallToothTime.push(this.gameStarThree-timeTwo);this.fallToothTime.push(this.gameStarThree-2\*timeTwo);this.fallToothTime.push(this.gameStarTwo);var timeThree=parseInt((this.gameStarTwo-this.gameStarOne)/3);this.fallToothTime.push(this.gameStarTwo-timeThree);this.fallToothTime.push(this.gameStarTwo-2\*timeThree);this.fallToothTime.push(this.gameStarOne);var timeFour=parseInt(this.gameStarOne/2);this.fallToothTime.push(this.gameStarOne-timeFour);this.fallToothTime.push(1);},checkAddTimeMonster:function(){var timeMonsterID=parseInt(this.score/5000);if(timeMonsterID!=this.addTimeMonsterID&&timeMonsterID>this.addTimeMonsterID){this.addTimeMonsterID=timeMonsterID;this.isCanAddTimeMonster=true;}},checkIsFullMonster:function(){var monsterCount=0;for(var i=3;i<Global.classicModeHeight-2;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null&&!monster.isShake){monsterCount++;}}}

if(monsterCount==63){return true;}else{return false;}},playTimeUPAnimation:function(){var timeUP=new cc.Sprite(res.timeUP\_png);timeUP.x=360;timeUP.y=640;this.addChild(timeUP,15);timeUP.setScale(0.1,0.1);var callFun=cc.callFunc(function(){timeUP.removeFromParent();});timeUP.runAction(cc.sequence(cc.scaleTo(0.2,1.15,1.15),cc.scaleTo(0.2,0.95,0.95),cc.scaleTo(0.2,1,1),cc.delayTime(0.2),callFun));},showGameTipsByID:function(ID){if(Global.currentMode==1){if(Global.currentSelectLevel==1){switch(ID){case 1:if(!currentScene.gameLayer.isShowTipsTwo){currentScene.gameLayer.isShowTipsTwo=true;var callFun=cc.callFunc(function(){currentScene.gameTips(2);});currentScene.runAction(cc.sequence(cc.delayTime(0.1),callFun));}

break;case 2:if(this.effectsType!=-1){if(Global.currentMode==1){if(Global.currentSelectLevel==1){if(!currentScene.gameLayer.isShowTipsThree){currentScene.gameLayer.isShowTipsThree=true;var callFun=cc.callFunc(function(){currentScene.gameTips(3);});currentScene.runAction(cc.sequence(cc.delayTime(0.1),callFun));}}}}

break;}}}},boomOneByOne:function(){var isExistMonster=false;for(var i=Global.classicModeHeight-2;i>=3;i--){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null&&!monster.isUp&&!monster.isDown&&!monster.isWaiteDie&&monster.isMonster&&monster.effectsType==-1&&!monster.isShake){this.monsterPlayBoom(monster);isExistMonster=true;break;}}

if(isExistMonster){break;}}},monsterPlayBoom:function(monster){monster.isWaiteDie=true;monster.visible=false;var sprite=this.createBoomEffects(monster,monster.skin+2);sprite.x=currentScene.gameLayer.tileMapPos[monster.rowcol[0]][monster.rowcol[1]].x+3;sprite.y=currentScene.gameLayer.tileMapPos[monster.rowcol[0]][monster.rowcol[1]].y+180;currentScene.addChild(sprite,5);},createBoomEffects:function(monster,effectsType){var arr=[];arr=Global.monsterDieEfects[effectsType];var sprite=new cc.Sprite("#"+arr[0]+"0000"+arr[1]+".png");var frames=[];for(var i=arr[1];i<=arr[2];i++){var frame=cc.spriteFrameCache.getSpriteFrame(arr[0]+"0000"+i+".png");frames.push(frame);}

var Animation=new cc.Animation.create(frames,0.07);var animate=new cc.Animate(Animation);var self=sprite;var myself=monster;var myLayer=this;var callFunAction=cc.callFunc(function(){self.removeFromParent();myLayer.boomOneByOne();});var callAction=cc.callFunc(function(){if(myself!=null&&myself.monster!=null){myself.removeMonster();}});sprite.runAction(cc.sequence(animate,callAction,callFunAction));return sprite;}});var gameTargetLayerTargetOffset=100;var GameTargetLayer=cc.Layer.extend({rootNode:null,gameTarget:null,countTime:0,score:0,gameStarOne:0,gameStarTwo:0,gameStarThree:0,gameTargetScore:0,gameTargetOneID:0,gameTargetOneNum:0,gameTargetTwoID:0,gameTargetTwoNum:0,gameInitFallCD:0,gameFallCD:0,ctor:function(){this.\_super();this.init();},init:function(){this.initGameTargetData();this.initUI();},initUI:function(){this.rootNode=loadCSS(res.gameTargetLayer\_json);this.addChild(this.rootNode);this.initGameTargetUI();this.rootNode.Image\_4.visible=false;this.rootNode.Image\_4.setScale(0.5,0.5);this.rootNode.Image\_4.runAction(cc.fadeOut(0.1));var self=this;self.setCascadeOpacityEnabled(true);var callFunAction=cc.callFunc(function(){self.rootNode.Image\_1.visible=false;self.rootNode.Image\_4.visible=true;playEffect(24);self.rootNode.Image\_4.runAction(cc.spawn(cc.scaleTo(0.2,1.2,1.2),cc.fadeIn(0.3),cc.scaleTo(0.1,1.0,1.0)));});var callFunActionII=cc.callFunc(function(){self.removeFromParent();var picID=-1;switch(Global.currentSelectLevel){case 1:picID=1;break;case 2:picID=4;break;case 3:picID=5;break;case 4:picID=6;break;case 5:picID=7;break;}

if(picID!=-1){currentScene.gameTips(picID);}

currentScene.gameLayer.scheduleFun();});this.rootNode.Image\_1.runAction(cc.sequence(cc.moveTo(0.3,cc.p(360,640)),cc.delayTime(1.5),callFunAction,cc.delayTime(1),callFunActionII));},initGameTargetData:function(){this.gameTarget=[];this.countTime=Global.gameTargetData[Global.currentSelectLevel-1][0];this.gameStarOne=Global.gameTargetData[Global.currentSelectLevel-1][1];this.gameStarTwo=Global.gameTargetData[Global.currentSelectLevel-1][2];this.gameStarThree=Global.gameTargetData[Global.currentSelectLevel-1][3];this.gameTargetScore=Global.gameTargetData[Global.currentSelectLevel-1][4];this.gameTargetOneID=Global.gameTargetData[Global.currentSelectLevel-1][5];this.gameTargetOneNum=Global.gameTargetData[Global.currentSelectLevel-1][6];this.gameTargetTwoID=Global.gameTargetData[Global.currentSelectLevel-1][7];this.gameTargetTwoNum=Global.gameTargetData[Global.currentSelectLevel-1][8];this.gameInitFallCD=Global.gameTargetData[Global.currentSelectLevel-1][9];this.gameFallCD=Global.gameTargetData[Global.currentSelectLevel-1][10];},initGameTargetUI:function(){if(this.gameTargetScore>0){this.gameTarget.push(11);this.gameTarget.push(this.gameTargetScore);this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[10]);this.rootNode.Image\_1.BitmapFontLabel\_2.setString(this.gameTargetScore+"");if(this.gameTargetOneID>0){this.gameTarget.push(this.gameTargetOneID);this.gameTarget.push(this.gameTargetOneNum);this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID-1]);this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x"+this.gameTargetOneNum);}else{if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x"+this.gameTargetTwoNum);}else{this.rootNode.Image\_1.Sprite\_3.visible=false;this.rootNode.Image\_1.BitmapFontLabel\_3.visible=false;this.rootNode.Image\_1.Sprite\_2.x+=gameTargetLayerTargetOffset;this.rootNode.Image\_1.BitmapFontLabel\_2.x+=gameTargetLayerTargetOffset;}}}else{if(this.gameTargetOneID>0){this.gameTarget.push(this.gameTargetOneID);this.gameTarget.push(this.gameTargetOneNum);this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[this.gameTargetOneID-1]);this.rootNode.Image\_1.BitmapFontLabel\_2.setString("x"+this.gameTargetOneNum);if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.Image\_1.Sprite\_3.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.Image\_1.BitmapFontLabel\_3.setString("x"+this.gameTargetTwoNum);}else{this.rootNode.Image\_1.Sprite\_3.visible=false;this.rootNode.Image\_1.BitmapFontLabel\_3.visible=false;this.rootNode.Image\_1.Sprite\_2.x+=gameTargetLayerTargetOffset;this.rootNode.Image\_1.BitmapFontLabel\_2.x+=gameTargetLayerTargetOffset;}}else{if(this.gameTargetTwoID>0){this.gameTarget.push(this.gameTargetTwoID);this.gameTarget.push(this.gameTargetTwoNum);this.rootNode.Image\_1.Sprite\_2.initWithFile(Global.gameTargetPicUrl[this.gameTargetTwoID-1]);this.rootNode.Image\_1.BitmapFontLabel\_2.setString("x"+this.gameTargetTwoNum);this.rootNode.Image\_1.Sprite\_3.visible=false;this.rootNode.Image\_1.BitmapFontLabel\_3.visible=false;this.rootNode.Image\_1.Sprite\_2.x+=gameTargetLayerTargetOffset;this.rootNode.Image\_1.BitmapFontLabel\_2.x+=gameTargetLayerTargetOffset;}else{this.rootNode.Image\_1.Sprite\_2.visible=false;this.rootNode.Image\_1.BitmapFontLabel\_2.visible=false;}}}}});var GamePauseLayer=cc.Layer.extend({rootNode:null,home:null,music:null,musicoff:null,help:null,continue:null,onEnter:function(){this.\_super();this.init();},init:function(){this.initUI();this.initText();this.initListener();},initText:function(){},initUI:function(){this.rootNode=loadCSS(res.GamePauseLayer\_json);this.addChild(this.rootNode);this.home=this.rootNode.Image\_2.Button\_1;this.home.tag=1;this.home.setPressedActionEnabled(true);this.music=this.rootNode.Image\_2.Button\_4;this.music.tag=2;this.music.setPressedActionEnabled(true);this.musicoff=this.rootNode.Image\_2.Button\_5;this.musicoff.tag=5;this.musicoff.setPressedActionEnabled(true);this.help=this.rootNode.Image\_2.Button\_2;this.help.tag=3;this.help.setPressedActionEnabled(true);this.continue=this.rootNode.Image\_2.Button\_3;this.continue.tag=4;this.continue.setPressedActionEnabled(true);this.initMusic();},initMusic:function(){if(gamePlayerData.soundSwitch){this.music.visible=true;this.musicoff.visible=false;}else{this.music.visible=false;this.musicoff.visible=true;}},initListener:function(){this.home.addTouchEventListener(this.btnCallFun);this.music.addTouchEventListener(this.btnCallFun);this.help.addTouchEventListener(this.btnCallFun);this.continue.addTouchEventListener(this.btnCallFun);this.musicoff.addTouchEventListener(this.btnCallFun);},btnCallFun:function(sender,type){switch(sender.tag){case 1:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("home");if(cc.director.isPaused()){cc.director.resume();currentScene.gameLayer.isPause=false;}

if(Global.currentMode==1){transitions(SelectScene);}

if(Global.currentMode==2){transitions(LoginScene);}

break;}

break;case 2:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("music");gamePlayerData.soundSwitch=false;stopBgMusic(1);currentScene.gamePauseLayer.initMusic();break;}

break;case 3:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("help");if(cc.director.isPaused()){cc.director.resume();currentScene.gameLayer.isPause=false;}

transitions(GamePlayScene);break;}

break;case 4:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("continue");if(currentScene.gamePauseLayer!=null){if(cc.director.isPaused()){cc.director.resume();currentScene.gameLayer.isPause=false;}

currentScene.gamePauseLayer.removeFromParent();currentScene.gamePauseLayer=null;}

break;}

break;case 5:switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("musicoff");gamePlayerData.soundSwitch=true;playBgMusic(1);currentScene.gamePauseLayer.initMusic();break;}

break;}

savePlayerData();}});var EndlessGameLayer=cc.Layer.extend({rootNode:null,btnPause:null,btnAddGold:null,btnToiletSucker:null,btnDyeBottle:null,panel:null,touchPanel:null,tileMap:null,tileMapx:null,tileMapPos:null,tileMapUp:null,tileMapDown:null,tempMonsterArr:null,tempMonsterArrs:null,removeRows:null,touchMonster:null,isCreateMonsterDown:false,isCreateCompleted:false,tempColsArr:null,isClearMonster:false,isTouchDown:false,touchPosition:null,gameControl:null,touchPosInBox:null,frontGrid:null,currentGrid:null,isGenerateMonster:false,connectMonsterArrs:null,isTouchInBox:false,isSelectDyeBottle:false,isDyeing:false,isSelectToiletSucker:false,isPlayToilet:false,isBanFall:false,countTime:0,score:0,gameStarOne:0,gameStarTwo:0,gameStarThree:0,gameTargetScore:0,gameTargetOneID:0,gameTargetOneNum:0,gameTargetTwoID:0,gameTargetTwoNum:0,gameInitFallCD:0,gameFallCD:0,gameTarget:null,isGameWin:false,isGameOver:false,isTouchOnMouseUp:false,LV:1,isShowGameAlarm:false,scoreBar:null,gameLevelUpNode:null,ctor:function(){this.\_super();this.init();this.scheduleOnce(this.scheduleFallMonster,10);},scheduleFallMonster:function(){this.createMonsterByRows(1,1);this.schedule(this.fallMonster,10);},onEnter:function(){this.\_super();this.schedule(this.updateCheckout,1);},init:function(){this.initUI();this.initData();this.initListener();},initGameTargetData:function(){this.gameTarget=[];this.countTime=Global.gameTargetData[Global.currentSelectLevel-1][0];this.gameStarOne=Global.gameTargetData[Global.currentSelectLevel-1][1];this.gameStarTwo=Global.gameTargetData[Global.currentSelectLevel-1][2];this.gameStarThree=Global.gameTargetData[Global.currentSelectLevel-1][3];this.gameTargetScore=Global.gameTargetData[Global.currentSelectLevel-1][4];this.gameTargetOneID=Global.gameTargetData[Global.currentSelectLevel-1][5];this.gameTargetOneNum=Global.gameTargetData[Global.currentSelectLevel-1][6];this.gameTargetTwoID=Global.gameTargetData[Global.currentSelectLevel-1][7];this.gameTargetTwoNum=Global.gameTargetData[Global.currentSelectLevel-1][8];this.gameInitFallCD=Global.gameTargetData[Global.currentSelectLevel-1][9];this.gameFallCD=Global.gameTargetData[Global.currentSelectLevel-1][10];},initData:function(){this.tempMonsterArrs=[];this.removeRows=[];this.connectMonsterArrs=[];this.tempColsArr=[0,0,0,0,0,0,0];this.tileMap=new TileMap();var tileMapData=this.createTileMapData(Global.classicModeOffsetX,Global.classicModeOffsetY,Global.tileMapWidth,Global.tileMapHeight,Global.classicModeWidth,Global.classicModeHeight,null);this.tileMap.initialize(tileMapData);this.createMapPoint();},createMapPoint:function(){this.tileMapPos=[];for(var row=0;row<Global.classicModeHeight;row++){var arr=[];for(var col=0;col<Global.classicModeWidth;col++){if(row==Global.classicModeHeight-1){var point=cc.p(col\*Global.tileMapWidth+Global.monsterWidth/2+43,Global.tileMapHeight\*(row-3)+Global.monsterHeight/2+30);}else{var point=cc.p(col\*Global.tileMapWidth+Global.monsterWidth/2+43,Global.tileMapHeight\*(row-3)+Global.monsterHeight/2);}

arr.push(point);}

this.tileMapPos.push(arr);}},createTileMapData:function(offsetX,offsetY,gridWidth,gridHeight,gridMaxX,gridMaxY,defaultGridValue){var data=new Object();data.offsetX=offsetX;data.offsetY=offsetY;data.gridWidth=gridWidth;data.gridHeight=gridHeight;data.gridMaxX=gridMaxX;data.gridMaxY=gridMaxY;data.defaultGridValue=defaultGridValue;return data;},initUI:function(){this.rootNode=loadCSS(res.EndlessGameLayer\_json);this.addChild(this.rootNode);this.btnPause=this.rootNode.Button\_1;this.btnAddGold=this.rootNode.Image\_8.Button\_2;this.btnAddGold.setPressedActionEnabled(true);this.btnDyeBottle=this.rootNode.Image\_9;this.rootNode.dyebottlex\_1.visible=false;this.btnToiletSucker=this.rootNode.Image\_12;this.rootNode.toiletsuckerx\_2.visible=false;this.rootNode.Image\_1.setLocalZOrder(-3);this.rootNode.GameTongue\_1.setLocalZOrder(-2);this.touchPanel=new cc.Layer();this.touchPanel.width=720;this.touchPanel.height=960;this.touchPanel.x=3;this.touchPanel.y=180;this.rootNode.addChild(this.touchPanel,-1);this.gameControl=currentScene.gameControl;this.rootNode.BitmapFontLabel\_1.setString(""+1);this.rootNode.BitmapFontLabel\_2.setString(""+0);this.rootNode.GameAlarm\_1.visible=false;this.refreshCurrentMoney();this.scoreBar=this.rootNode.Image\_14.LoadingBar\_2;this.scoreBar.setPercent(0);this.refreshBtnAddMoney();this.gameLevelUpNode=loadCSS(res.GameLevelUpNode\_json);this.gameLevelUpNode.x=360;this.gameLevelUpNode.y=1920;this.addChild(this.gameLevelUpNode,20);this.showGameLevelUpNode();},refreshBtnAddMoney:function(){if(gamePlayerData.isJiHuo==0){this.btnAddGold.visible=false;}

if(gamePlayerData.isJiHuo==1){this.btnAddGold.visible=true;}},refreshCurrentLV:function(){this.rootNode.BitmapFontLabel\_1.setString(""+this.LV);},refreshCurrentScore:function(recordConnectRemoveTimes,removeCount){if(!this.isGameOver){var addScore=(recordConnectRemoveTimes\*100+300)\*(removeCount-2);this.score+=addScore;this.rootNode.BitmapFontLabel\_2.setString(this.score+"");currentScene.createEncourage(addScore);}

this.updateLVAndTime(this.score);},refreshCurrentMoney:function(){this.rootNode.Image\_8.BitmapFontLabel\_3.setString(""+gamePlayerData.money);},updateLVAndTime:function(score){var Lv=1;if(score<10000){Lv=1;this.refreshScoreBar(0,10000);}

if(score>=10000&&score<25000){Lv=2;this.refreshScoreBar(10000,25000);}else if(score>=25000&&score<40000){Lv=3;this.refreshScoreBar(25000,40000);}else if(score>=40000&&score<58000){Lv=4;this.refreshScoreBar(40000,58000);}else if(score>=58000&&score<70000){Lv=5;this.refreshScoreBar(58000,70000);}else if(score>=70000&&score<95000){Lv=6;this.refreshScoreBar(70000,95000);}else if(score>=95000&&score<115000){Lv=7;this.refreshScoreBar(95000,115000);}else if(score>=115000&&score<125000){Lv=8;this.refreshScoreBar(115000,125000);}else if(score>=125000&&score<150000){Lv=9;this.refreshScoreBar(125000,150000);}else if(score>=150000){Lv=parseInt(score-150000)/20000+10;this.refreshScoreBar(150000+(Lv-10)\*20000,150000+(Lv-10)\*20000+20000);}else{cc.log("score < 10000");}

if(this.LV==Lv){cc.log("this.Lv == Lv");}else{this.LV=Lv;this.refreshCurrentLV();this.unschedule(this.fallMonster);if(Lv<=9){var fallCD=Global.fallCD[Lv-2];this.schedule(this.fallMonster,fallCD);}else{this.schedule(this.fallMonster,3);}

this.showGameLevelUpNode();}},refreshScoreBar:function(scoreMin,scoreMax){var value=(this.score-scoreMin)/(scoreMax-scoreMin)\*100;this.scoreBar.setPercent(value);},refreshCurrentScoreByScore:function(score){this.score+=score;this.rootNode.BitmapFontLabel\_2.setString(this.score+"");this.updateLVAndTime(this.score);},initListener:function(){this.btnPause.addTouchEventListener(this.btnPauseFun,this.btnPause);this.btnAddGold.addTouchEventListener(this.btnAddGoldFun,this.btnAddGold);engine.gameEvent.createMouseDown(this.btnDyeBottle,this.btnDyeBottleFun);engine.gameEvent.createMouseDown(this.btnToiletSucker,this.btnToiletSuckerFun);this.addTouchListener();},btnDyeBottleFun:function(){playEffect(2);cc.log("btnDyeBottleFun");if(currentScene.gameLayer.isSelectToiletSucker)return;if(currentScene.gameLayer.isSelectDyeBottle){if(currentScene.gameLayer.isDyeing){cc.log("正在染色");}else{currentScene.gameLayer.isSelectDyeBottle=false;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=false;}}else{if(gamePlayerData.money>=500){currentScene.gameLayer.isSelectDyeBottle=true;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=true;}}

savePlayerData();},btnToiletSuckerFun:function(){playEffect(2);cc.log("btnToiletSuckerFun");if(currentScene.gameLayer.isSelectDyeBottle)return;if(currentScene.gameLayer.isSelectToiletSucker){if(currentScene.gameLayer.isPlayToilet){cc.log("正在播放马桶塞动画");}else{currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;}}else{if(gamePlayerData.money>=200){currentScene.gameLayer.isSelectToiletSucker=true;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=true;}}

savePlayerData();},btnPauseFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(4);cc.log("btnPauseFun");if(!this.isGameOver&&!this.isGameWin){this.isPause=true;cc.director.pause();currentScene.gamePause();}

break;}},btnAddGoldFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnAddGoldFun");createSDKLayerFun();break;}},initMap:function(){this.upInitMonster();},createMonster:function(i,j,monsterType){if(monsterType<=0)return;var monster=new SpecialMonsterNode(monsterType,i,j);if(monster!=null){monster.x=this.tileMapPos[i][j].x;monster.y=this.tileMapPos[i][j].y;this.touchPanel.addChild(monster,Global.classicModeHeight-(i-3));monster.rowcol=[i,j];monster.row=monster.rowcol[0];this.tileMap.setGridInfo(i,j,monster);}},fallMonstersCallFun:function(){this.printArr();this.searchConnectedMonsters(this.tileMap);cc.log("//判断是否有相连（3个以上)");var isConnected=this.checkIsConnectedMonsters();if(isConnected){if(this.touchMonster==null){if(!this.isTouchDown){var isFallDowaning=this.searchIsFallDown(this.tileMap);var isMonsterShake=this.searchIsShakeMonster(this.tileMap);cc.log(this.tileMap.gridArr);if(isFallDowaning&&isMonsterShake){this.upInitMonster();}}}}else{this.searchConnectedMonstersArrs();this.shakeConnectedMonsters();}},checkIsConnectedMonsters:function(){if(this.tempMonsterArrs==null)return;var connectedMonstersTimes=0;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){connectedMonstersTimes++;}}}}

if(connectedMonstersTimes==0){return true;}else{return false;}},createMonsterByRows:function(rows,upAndDown){var monsterTypeArr=[];for(var j=0;j<rows;j++){for(var i=0;i<7;i++){var monsterType=randomNums(1,9);if(monsterType==7){monsterType=6;}

if(this.LV<5){if(monsterType==6){monsterType=randomNums(1,6)}

if(monsterType==8){monsterType=randomNums(1,6);}}

if(this.LV>=5&&this.LV<7){if(monsterType==8){monsterType=randomNums(1,6);}}

if(i>=2){if(monsterTypeArr[i-1]==monsterType&&monsterTypeArr[i-2]==monsterType){if(monsterType!=1){monsterType=monsterType-1;}else{monsterType=2;}}}

monsterTypeArr.push(monsterType);if(upAndDown==1){if(this.tileMap.gridArr[Global.classicModeHeight-1][i]==null&&this.tileMap.gridArr[Global.classicModeHeight-2][i]==null){var monster=new SpecialMonsterNode(monsterType,Global.classicModeHeight-1,i);monster.x=this.tileMapPos[Global.classicModeHeight-1][i].x;monster.y=this.tileMapPos[Global.classicModeHeight-1][i].y;monster.isCanFallDown=false;this.touchPanel.addChild(monster);monster.rowcol=[Global.classicModeHeight-1,i];this.tileMap.setGridInfo(Global.classicModeHeight-1,i,monster);}}

if(upAndDown==2){if(this.tileMap.gridArr[3-1-j+1][i]==null){var monster=new SpecialMonsterNode(monsterType,3-1-j+1,i);monster.x=this.tileMapPos[3-1-j][i].x;monster.y=this.tileMapPos[3-1-j][i].y;monster.isCanFallDown=true;this.touchPanel.addChild(monster);this.tileMap.setGridInfo(3-1-j,i,monster);this.gameControl.upMonster(3-1-j,i,monster,rows);}}}}},monsterCallFun:function(target,data){if(data==null)return;var monster=data[0];var j=data[1];var i=data[2];monster.rowcol=[j,i];monster.setLocalZOrder(Global.classicModeHeight-(j-3));monster.row=monster.rowcol[0];cc.log("monster.rowcol :"+monster.rowcol);this.tileMap.setGridInfo(j,i,monster);},searchRowsOfTileMap:function(tileMap,isFull){if(tileMap==null)return;var tempNum=0;var num=0;if(isFull){for(var i=3;i<tileMap.gridMaxY-1;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){num++;}

if(j==tileMap.gridMaxX-1){if(num==tileMap.gridMaxX){tempNum++;}

num=0;}}}}else{for(var \_i=3;\_i<tileMap.gridMaxY-1;\_i++){for(var \_j=0;\_j<tileMap.gridMaxX;\_j++){if(tileMap.gridArr[\_i][\_j]!=null){tempNum++;break;}}}}

return tempNum;},searchColsOfTileMap:function(tileMap){if(tileMap==null)return;var tempNum=[];for(var i=0;i<tileMap.gridMaxX;i++){if(tileMap.gridArr[tileMap.gridMaxY-1-1][i]!=null){tempNum.push(i);}}

return tempNum;},searchConnectedMonsters:function(tileMap){this.tempMonsterArrs=[];var arr=[];for(var i=3;i<tileMap.gridMaxY-2;i++){for(var j=0;j<tileMap.gridMaxX;j++){this.tempMonsterArr=[];if(tileMap.gridArr[i][j]!=null){if(tileMap.gridArr[i][j].isMonster||tileMap.gridArr[i][j].isTimeMonster||tileMap.gridArr[i][j].isCageMonster||tileMap.gridArr[i][j].isBoxMonster||tileMap.gridArr[i][j].isAdvancedBoxMonster||tileMap.gridArr[i][j].isMonsterKing){if(!tileMap.gridArr[i][j].isSearched&&!tileMap.gridArr[i][j].isWaiteDie&&!tileMap.gridArr[i][j].isDown&&!tileMap.gridArr[i][j].isUp){if(i==3||tileMap.gridArr[i-1][j]!=null){tileMap.gridArr[i][j].isSearched=true;this.tempMonsterArr.push(tileMap.gridArr[i][j]);this.searchMonstersByDir(i,j,tileMap,0,1);this.searchMonstersByDir(i,j,tileMap,0,-1);this.searchMonstersByDir(i,j,tileMap,-1,0);this.searchMonstersByDir(i,j,tileMap,1,0);this.tempMonsterArrs.push(this.tempMonsterArr);}}}}}}

for(var i=3;i<tileMap.gridMaxY-2;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){tileMap.gridArr[i][j].isSearched=false;}}}},searchMonstersByDir:function(i,j,tileMap,horizontal,vertical){if(i+vertical>2&&i+vertical<13){if(tileMap.isExistGrid(j+horizontal,i+vertical)){if(tileMap.gridArr[i+vertical][j+horizontal]!=null){if(tileMap.gridArr[i+vertical][j+horizontal].isMonster||tileMap.gridArr[i+vertical][j+horizontal].isTimeMonster||tileMap.gridArr[i+vertical][j+horizontal].isCageMonster||tileMap.gridArr[i+vertical][j+horizontal].isBoxMonster||tileMap.gridArr[i+vertical][j+horizontal].isAdvancedBoxMonster||tileMap.gridArr[i+vertical][j+horizontal].isMonsterKing){if(tileMap.gridArr[i+vertical][j+horizontal].skin==tileMap.gridArr[i][j].skin||tileMap.gridArr[i+vertical][j+horizontal].isTimeMonster){if(!tileMap.gridArr[i+vertical][j+horizontal].isSearched&&!tileMap.gridArr[i+vertical][j+horizontal].isWaiteDie&&!tileMap.gridArr[i+vertical][j+horizontal].isDown&&!tileMap.gridArr[i+vertical][j+horizontal].isUp){if(i+vertical==3||tileMap.gridArr[i+vertical-1][j+horizontal]!=null){tileMap.gridArr[i+vertical][j+horizontal].isSearched=true;this.tempMonsterArr.push(tileMap.gridArr[i+vertical][j+horizontal]);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,0,1);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,0,-1);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,1,0);this.searchMonstersByDir(i+vertical,j+horizontal,tileMap,-1,0);}}}}}}}},searchConnectedMonstersArrs:function(){if(this.tempMonsterArrs==null)return;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){this.connectMonsterArrs.push(this.tempMonsterArrs[i]);}}}}},shakeConnectedMonsters:function(){if(this.connectMonsterArrs==null)return;var connectMonsterArrsLength=this.connectMonsterArrs.length;for(var i=0;i<connectMonsterArrsLength;i++){var plength=this.connectMonsterArrs[i].length;if(plength>=3){var skin=this.connectMonsterArrs[i][0].skin;cc.log("shake 相连的怪物");var isExistArr=false;var shakeTimes=Global.monsterShakeTimes;for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(this.connectMonsterArrs[i][j].isShake){var removeAct=this.gameControl.searchExistArrFromRemoveList(this.connectMonsterArrs[i][j]);isExistArr=true;break;}}

if(isExistArr){for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(!this.connectMonsterArrs[i][j].isShake){if(removeAct!=null){if(removeAct.arr!=null){this.connectMonsterArrs[i][j].isShake=true;playEffect(19);this.connectMonsterArrs[i][j].playShakeAni();removeAct.times=removeAct.times+5;removeAct.arr.push(this.connectMonsterArrs[i][j]);playEffect(20);}}}}}else{var isMonsterRemove=false;for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(this.connectMonsterArrs[i][j].isRemve){this.connectMonsterArrs[i][j].isRemve=false;isMonsterRemove=true;}}

for(var j=0;j<plength;j++){if(this.connectMonsterArrs[i][j]==null)return;if(!this.connectMonsterArrs[i][j].isShake){this.connectMonsterArrs[i][j].isShake=true;this.connectMonsterArrs[i][j].playShakeAni();if(j==plength-1){playEffect(19);}}}

this.gameControl.createRemoveList(this.connectMonsterArrs[i],shakeTimes,skin);}}}

this.connectMonsterArrs=[];},removeConnectedMonsters:function(){if(this.tempMonsterArrs==null)return;this.tempColsArr=[0,0,0,0,0,0,0];this.isClearMonster=true;var tempMonsterArrLength=this.tempMonsterArrs.length;if(this.tempMonsterArrs.length>0){for(var i=0;i<tempMonsterArrLength;i++){if(this.tempMonsterArrs[i]!=null){var pLength=this.tempMonsterArrs[i].length;if(pLength>=3){cc.log("移出相连的怪物");}}}}

this.tempMonsterArr=[];this.tempMonsterArrs=[];},fallMonstersBySpace:function(){Global.monsterDown=false;if(this.isBanFall)return;this.setFallRemoveRows();for(var i=0;i<this.removeRows.length;i++){var row=this.removeRows[i];var space=0;for(var j=3;j<Global.classicModeHeight;j++){var monster=this.tileMap.gridArr[j][row];if(monster==null||monster.isBox||monster.isAdvancedBox){space++;}else{if(space>0){if(!monster.isShake&&!monster.isDown&&!monster.isWaiteDie&&monster.isCanFallDown){if(monster.isMonster||monster.isIce||monster.isTimeMonster||monster.isCageMonster||monster.isMonsterKing){this.gameControl.moveMonster(j,row,monster,1);}}

if(monster.isShake||monster.isBoxMonster||monster.isAdvancedBoxMonster||monster.isWaiteDie){space=0;}}}}}

this.removeRows=[];if(!Global.monsterDown){if(this.gameControl.isMoveEnd){if(currentScene.gameLayer.isTouchOnMouseUp){currentScene.gameLayer.isTouchOnMouseUp=false;}

cc.log("this.gameControl.isMoveEnd");var self=this;var callAction=cc.callFunc(function(){self.fallMonstersCallFun();});this.runAction(cc.sequence(cc.delayTime(0.2),callAction));}else{if(currentScene.gameLayer.isTouchOnMouseUp){currentScene.gameLayer.isTouchOnMouseUp=false;cc.log("currentScene.gameLayer.isTouchOnMouseUp +++++++++++++++++++++++ this.gameControl.isMoveEnd");var self=this;var callAction=cc.callFunc(function(){self.fallMonstersCallFun();});this.runAction(cc.sequence(cc.delayTime(0.2),callAction));}}}},fallMonster:function(){cc.log("每隔一段时间下落 一排怪物");if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver)return;var self=this;var callFun=cc.callFunc(function(){self.fallMonstersBySpace();});var callFunI=cc.callFunc(function(){self.createMonsterByRows(1,1);});var callFunII=cc.callFunc(function(){self.setMonsterCanFallDown();});var callFunIII=cc.callFunc(function(){self.playMonsterShake();});var callFunIIII=cc.callFunc(function(){self.stopMonsterShake();});this.runAction(cc.sequence(callFunIII,cc.delayTime(0.5),callFunIIII,callFunII,callFun,cc.delayTime(0.5),callFunI));},setMonsterCanFallDown:function(){for(var i=0;i<7;i++){var monster=this.tileMap.gridArr[Global.classicModeHeight-1][i];if(monster!=null){monster.isCanFallDown=true;}}},playMonsterShake:function(){for(var i=0;i<7;i++){var monster=this.tileMap.gridArr[Global.classicModeHeight-1][i];if(monster!=null){var shake=monsterShakeAction(monster.rowcol);var shakeAction=cc.sequence(shake[0],shake[1],shake[2],shake[3],shake[4],shake[5],shake[6],shake[7],shake[8]).repeatForever();shakeAction.setTag(5000);monster.runAction(shakeAction);}}},stopMonsterShake:function(){for(var i=0;i<7;i++){var monster=this.tileMap.gridArr[Global.classicModeHeight-1][i];if(monster!=null){monster.stopActionByTag(5000);}}},upInitMonster:function(){if(currentScene.gameLayer.isTouchDown||currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver)return;if(currentScene.gameControl.moveList.length!=0||currentScene.gameControl.removeList.length!=0||currentScene.gameControl.removeSpecialMonsterList.length!=0||currentScene.gameControl.upMonsterList.length>1||currentScene.gameControl.downMonsterList.length!=0){if(currentScene.gameControl.removeSpecialMonsterList.length!=0||currentScene.gameControl.removeList.length!=0){return;}else{return;}}

if(this.isCreateMonsterDown)return;var MonsterRowsCols=this.computeCreateMonsterDownRows(this.tileMap);var rows=MonsterRowsCols[0];if(rows==0){cc.log("upInitMonster");this.printArr();this.searchConnectedMonsters(this.tileMap);var isConnected=this.checkIsConnectedMonsters();if(!isConnected){this.searchConnectedMonstersArrs();this.shakeConnectedMonsters();}

return;}

this.isCreateMonsterDown=true;var cols=MonsterRowsCols[1];var tileMapRows=this.searchRowsOfTileMap(this.tileMap,false);for(var i=tileMapRows+3;i>=3;i--){for(var j=0;j<Global.classicModeWidth;j++){if(i<Global.classicModeHeight-1&&i+1<=Global.classicModeHeight-2){if(this.tileMap.gridArr[i+1][j]==null){var tileMapMonster=this.tileMap.gridArr[i][j];if(tileMapMonster!=null&&!tileMapMonster.isBox&&!tileMapMonster.isBoxMonster&&!tileMapMonster.isAdvancedBox&&!tileMapMonster.isAdvancedBoxMonster){if(!tileMapMonster.isDown&&!tileMapMonster.isShake&&!tileMapMonster.isWaiteDie&&!tileMapMonster.isUp){this.gameControl.upMonster(i,j,tileMapMonster,rows);}}}}}}

this.createMonsterByRows(rows,2);},getPosByGrid:function(Grid){return this.tileMapPos[Grid[1]][Grid[0]];},addTouchListener:function(){engine.gameEvent.createMouseDown(this.touchPanel,this.onMouseDown);engine.gameEvent.createMouseMove(this.touchPanel,this.onMouseMove);engine.gameEvent.createMouseUp(this.touchPanel,this.onMouseUp);},onMouseDown:function(touch,event){currentScene.gameLayer.isTouchDown=true;if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver||currentScene.gameLayer.isBanFall)return;cc.log("onTouchBegan");var pos=touch.getLocation();if(currentScene.gameLayer.touchMonster==null){currentScene.gameLayer.selectTouchMonster(pos);}

return true;},selectTouchMonster:function(pos){currentScene.gameLayer.touchPosition=pos;if(!currentScene.gameLayer.checkTouchArea(pos))return;cc.log("上升时，不能移动+++++++");cc.log("上升时，不能移动-------");var tilePos=currentScene.gameLayer.getTileMapPosByPoint(pos);var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,tilePos);grid[1]=grid[1]+3;if((currentScene.gameLayer.isSelectDyeBottle||currentScene.gameLayer.isSelectToiletSucker)&&!currentScene.gameLayer.isDyeing&&!currentScene.gameLayer.isPlayToilet){if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]==null&&currentScene.gameLayer.isSelectToiletSucker){currentScene.gameLayer.isPlayToilet=false;currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;}else{var props=new PropsNode();if(currentScene.gameLayer.isSelectDyeBottle){props.type=1;currentScene.gameLayer.isDyeing=true;props.initProps(res.dyeingbottle\_json,"dyeingbottle",currentScene.gameLayer,grid);}

if(currentScene.gameLayer.isSelectToiletSucker){props.type=2;currentScene.gameLayer.isPlayToilet=true;props.initProps(res.toiletplug\_json,"toiletplug",currentScene.gameLayer,grid);}

var tempPos=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]];props.setPosition(tempPos);props.x=props.x+3;props.y=props.y+180;currentScene.gameLayer.addChild(props,50);}}else{currentScene.gameLayer.frontGrid=grid;if(!currentScene.gameLayer.checkMonsterMove(grid)){currentScene.gameLayer.isTouchDown=true;if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){var monster=currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]],1,tilePos);currentScene.gameLayer.touchMonster=monster;}else if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){var monster=currentScene.gameLayer.removedFromBox(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]],2,tilePos);currentScene.gameLayer.touchMonster=monster;}else{if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDown&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isUp&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isShake){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=true;currentScene.gameLayer.touchMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];}}

if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;return;}

if(currentScene.gameLayer.touchMonster.isMonster||currentScene.gameLayer.touchMonster.isTimeMonster||currentScene.gameLayer.touchMonster.isMonsterKing){if(currentScene.gameLayer.touchMonster.isFalling){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;return;}

if(currentScene.gameLayer.touchMonster.isShake)currentScene.gameLayer.touchMonster=null;if(currentScene.gameLayer.touchMonster!=null){currentScene.gameLayer.touchMonster.isDown=false;currentScene.gameLayer.touchMonster.isMove=true;if(!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox&&!currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox){currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]=null;}

if(currentScene.gameLayer.removeRows.indexOf(grid[0])==-1){currentScene.gameLayer.removeRows.push(grid[0]);}}}}}},onMouseMove:function(touch,event){if(currentScene.gameLayer.isGameWin||currentScene.gameLayer.isGameOver||currentScene.gameLayer.isBanFall)return;cc.log("onTouchMoved");var pos=touch.getLocation();if(!currentScene.gameLayer.checkTouchArea(pos))return;if(currentScene.gameLayer.touchMonster==null){currentScene.gameLayer.selectTouchMonster(pos);}

var delta=touch.getDelta();var tilePos=currentScene.gameLayer.getTileMapPosByPoint(pos);if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster)return;var currentPoint=currentScene.gameLayer.touchMonster.getPosition();var deltaX=tilePos.x-currentPoint.x;var deltaY=tilePos.y-currentPoint.y;var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,tilePos);grid[1]=grid[1]+3;currentScene.gameLayer.currentGrid=grid;if(currentScene.gameLayer.touchPosition!=null){if((currentScene.gameLayer.currentGrid[0]!=currentScene.gameLayer.frontGrid[0]||currentScene.gameLayer.currentGrid[1]!=currentScene.gameLayer.frontGrid[1])&&(Math.abs(tilePos.x-currentScene.gameLayer.touchPosition.x)>Global.tileMapWidth)){currentScene.gameLayer.touchPosition=null;currentScene.gameLayer.fallMonstersBySpace();}}

var monster=null;if(grid[0]>=0&&grid[0]<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-1){monster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]];}

var dirMonster=null;if(currentScene.gameLayer.checkOutScreen(tilePos)){if(currentScene.gameLayer.checkMonsterMove(grid))return;if(monster!=null){currentScene.gameLayer.isTouchInBox=true;var tilePosDeltaX=tilePos.x-currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;if(Math.abs(tilePosDeltaX)<=15){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-1||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(dirMonster!=null){cc.log("上左右 都已满");}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}

if(tilePosDeltaX<0&&Math.abs(tilePosDeltaX)>15&&Math.abs(tilePosDeltaX)<45){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-1||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(dirMonster!=null){cc.log("上左右 都已满");}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}

if(tilePosDeltaX>0&&Math.abs(tilePosDeltaX)>15&&Math.abs(tilePosDeltaX)<45){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1];if(dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]];if(grid[1]+1>=Global.classicModeHeight-1||dirMonster!=null){dirMonster=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1];if(grid[0]-1<0||dirMonster!=null){cc.log("上左右 都已满");}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]].y;}}}else{if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1].y;}}}

currentScene.gameLayer.touchMonster.setPosition(tilePos);}else{if(currentScene.gameLayer.isTouchInBox){currentScene.gameLayer.isTouchInBox=false;if(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]!=null){tilePos.x=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].x;tilePos.y=currentScene.gameLayer.tileMapPos[grid[1]][grid[0]].y;}}else{if(deltaX>0){if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,1,0)){tilePos.x=currentPoint.x;}}else{if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,-1,0)){tilePos.x=currentPoint.x;}}

if(deltaY>0){if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,0,1)){tilePos.y=currentPoint.y;}}else{if(currentScene.gameLayer.checkMonsterByDirPos(currentPoint,0,-1)){tilePos.y=currentPoint.y;}}}

currentScene.gameLayer.touchMonster.setPosition(tilePos);}}},onMouseUp:function(touch,event){cc.log("onTouchEnded");if(currentScene.gameLayer.touchMonster==null)return;if(currentScene.gameLayer.touchMonster.isIce||currentScene.gameLayer.touchMonster.isCage||currentScene.gameLayer.touchMonster.isBox||currentScene.gameLayer.touchMonster.isAdvancedBox||currentScene.gameLayer.touchMonster.isCageMonster){currentScene.gameLayer.touchMonster=null;currentScene.gameLayer.currentGrid=null;currentScene.gameLayer.frontGrid=null;return;}

var currentPoint=currentScene.gameLayer.touchMonster.getPosition();var grid=currentScene.gameLayer.getTileMapGirdByPoint(currentScene.gameLayer.tileMap,currentPoint);grid[1]=grid[1]+3;var gridPos=currentScene.gameLayer.getPosByGrid(grid);if(currentScene.gameLayer.touchMonster!=null){currentScene.gameLayer.isTouchDown=false;currentScene.gameLayer.touchMonster.isMove=false;currentScene.gameLayer.touchMonster.setPosition(gridPos);if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]==null){cc.log("onMouseUp——该格子为空");currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{cc.log("onMouseUp——该格子不为空");if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox){var skin=currentScene.gameLayer.touchMonster.skin;currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin+1);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin=skin;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBox=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isBoxMonster=true;}else if(currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]]!=null&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox){var skin=currentScene.gameLayer.touchMonster.skin;currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].createBasisMonster(skin+1);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].skin=skin;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBox=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol=[grid[1],grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]].isAdvancedBoxMonster=true;}else{cc.log("该格子 已被占用");if(grid[0]-1>=0&&grid[0]-1<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-1&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]-1]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].rowcol=[grid[1],grid[0]-1];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]-1].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{if(grid[0]+1>=0&&grid[0]+1<Global.classicModeWidth&&grid[1]>=3&&grid[1]<Global.classicModeHeight-1&&currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]][grid[0]+1]);currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].rowcol=[grid[1],grid[0]+1];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].row=currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]][grid[0]+1].setLocalZOrder(Global.classicModeHeight-(grid[1]-3));}else{if(grid[0]>=0&&grid[0]<Global.classicModeWidth&&grid[1]+1>=3&&grid[1]+1<Global.classicModeHeight-1&&currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]]==null){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[grid[1]+1][grid[0]]);currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].isDelete=false;currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].rowcol=[grid[1]+1,grid[0]];currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].row=currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].rowcol[0];currentScene.gameLayer.tileMap.gridArr[grid[1]+1][grid[0]].setLocalZOrder(Global.classicModeHeight-(grid[1]+1-3));}else{var isHaveEmptyGrid=false;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){if(currentScene.gameLayer.tileMap.gridArr[i][j]==null&&!isHaveEmptyGrid){currentScene.gameLayer.touchMonster.setPosition(currentScene.gameLayer.tileMapPos[i][j]);currentScene.gameLayer.tileMap.gridArr[i][j]=currentScene.gameLayer.touchMonster;currentScene.gameLayer.tileMap.gridArr[i][j].isDelete=false;currentScene.gameLayer.tileMap.gridArr[i][j].rowcol=[i,j];currentScene.gameLayer.tileMap.gridArr[i][j].row=currentScene.gameLayer.tileMap.gridArr[i][j].rowcol[0];currentScene.gameLayer.tileMap.gridArr[i][j].setLocalZOrder(Global.classicModeHeight-(i+1-3));isHaveEmptyGrid=true;}}}

if(!isHaveEmptyGrid){currentScene.gameLayer.touchMonster.removeFromParent();currentScene.gameLayer.touchMonster=null;}}}}}}}

currentScene.gameLayer.isTouchOnMouseUp=true;currentScene.gameLayer.fallMonstersBySpace();currentScene.gameLayer.touchMonster=null;currentScene.gameLayer.currentGrid=null;currentScene.gameLayer.frontGrid=null;currentScene.gameLayer.isTouchDown=false;},setFallRemoveRows:function(){this.removeRows=[];for(var i=0;i<Global.classicModeWidth;i++){this.removeRows.push(i);}},checkTouchArea:function(pos){if(pos.x<718&&pos.x>2&&pos.y<960&&pos.y>180){return true;}else{return false;}},getTileMapGirdByPoint:function(tileMap,pos){var posX=pos.x-43;var posY=pos.y;var lx=parseInt(posX/tileMap.gridWidth);var ly=parseInt(posY/tileMap.gridHeight);return[lx,ly];},getTileMapPosByPoint:function(pos){return cc.p(pos.x-this.touchPanel.x,pos.y-this.touchPanel.y);},checkOutScreen:function(pos){if(pos.x-Global.monsterWidth/2>25&&pos.y-Global.monsterHeight/2>5&&pos.x+Global.monsterWidth/2<Global.classicModeWidth\*Global.tileMapWidth+50&&pos.y+Global.monsterHeight/2<Global.classicModeHeight\*Global.tileMapHeight+5){return true;}else{return false;}},checkMonsterByDirPos:function(pos,horizontal,vertical){var lx=parseInt((pos.x+horizontal-43+horizontal\*this.tileMap.gridWidth/2)/this.tileMap.gridWidth);var ly=parseInt((pos.y+vertical+vertical\*this.tileMap.gridHeight/2)/this.tileMap.gridHeight);ly=ly+3;if(this.tileMap.gridArr[ly][lx]==null){return false;}else{return true;}},checkMonsterMove:function(gird){var bool1=this.checkGrid(gird[0],gird[1]-1);var bool2=this.checkGrid(gird[0]-1,gird[1]);var bool3=this.checkGrid(gird[0]+1,gird[1]);var bool4=this.checkGrid(gird[0],gird[1]+1);var bool=bool1+bool2+bool3+bool4;if(bool==4){return true;}else{return false;}},checkGrid:function(gird0,gird1){if(gird0>=0&&gird0<Global.classicModeWidth&&gird1>=3&&gird1<Global.classicModeHeight-1){if(this.tileMap.gridArr[gird1][gird0]!=null){return 1;}else{return 0;}}else{return 1;}},computeCreateMonsterDownRows:function(tileMap){if(tileMap==null)return;var allMonsterNum=0;var rows=0;for(var i=3;i<tileMap.gridMaxY-1;i++){for(var j=0;j<tileMap.gridMaxX;j++){if(tileMap.gridArr[i][j]!=null){allMonsterNum++;}}}

if(allMonsterNum>32){rows=0;}else{if(allMonsterNum>=21&&allMonsterNum<28){rows=0;}

if(allMonsterNum>=14&&allMonsterNum<21){rows=1;}

if(allMonsterNum>=7&&allMonsterNum<14){rows=1;}

if(allMonsterNum<7){rows=1;}}

var cols=this.searchColsOfTileMap(tileMap);return[rows,cols];},searchIsFallDown:function(tileMap){var times=0;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=tileMap.gridArr[i][j];if(monster!=null){if(monster.isFalling){times++;}}}}

if(times==0){return true;}else{return false;}},searchIsShakeMonster:function(tileMap){var times=0;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=tileMap.gridArr[i][j];if(monster!=null){if(monster.isShake){times++;}}}}

if(times==0){return true;}else{return false;}},printArr:function(){for(var i=Global.classicModeHeight-1;i>2;i--){var temp="";for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null){switch(monster.skin){case 0:temp=temp+""+"蓝";break;case 1:temp=temp+""+"绿";break;case 2:temp=temp+""+"紫";break;case 3:temp=temp+""+"红";break;case 4:temp=temp+""+"黄";break;}}else{temp=temp+""+0;}}

cc.log(temp);}},starlightMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;for(var i=-1;i<2;i++){for(var j=-1;j<2;j++){var tempMonster=this.tileMap.gridArr[colRow[0]+i][colRow[1]+j];if(colRow[0]+i>=3&&colRow[0]+i<Global.classicModeHeight-1&&colRow[1]+j>=0&&colRow[1]+j<Global.classicModeWidth&&tempMonster!=null){if(i==0&&j==0){cc.log("自己已经被删除");}else{if(!tempMonster.isShake&&!tempMonster.isWaiteDie&&!tempMonster.isDown&&!tempMonster.isUp){playEffect(22);tempMonster.initDieEffectsType(1);if(tempMonster.effectsType==1){tempMonster.flameMonsterDie();}else{tempMonster.playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(tempMonster);}}}}}},flameMonsterFall:function(monster){if(monster==null)return;var colRow=monster.rowcol;if(monster.row==3){cc.log("移到底炸周边六个");monster.playDieAni();}else{if(this.tileMap.gridArr[monster.row-1][colRow[1]]!=null&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isShake&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isWaiteDie&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isDown&&!this.tileMap.gridArr[monster.row-1][colRow[1]].isUp){playEffect(22);this.tileMap.gridArr[monster.row-1][colRow[1]].initDieEffectsType(6);if(this.tileMap.gridArr[monster.row-1][colRow[1]].effectsType==1){this.tileMap.gridArr[monster.row-1][colRow[1]].flameMonsterDie();}else{this.tileMap.gridArr[monster.row-1][colRow[1]].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[monster.row-1][colRow[1]]);}

var moveAction=cc.moveBy(0.03,cc.p(0,-90));var self=this;var callBackAction=cc.callFunc(function(){monster.row=monster.row-1;self.flameMonsterFall(monster);});monster.runAction(cc.sequence(cc.delayTime(0.05),moveAction,callBackAction));}},flameMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;var row=monster.row;if(row==3){cc.log("移到底炸周边六个");if(colRow[1]-1>=0&&colRow[1]-1<Global.classicModeWidth&&this.tileMap.gridArr[row][colRow[1]-1]!=null){if(!this.tileMap.gridArr[row][colRow[1]-1].isWaiteDie&&!this.tileMap.gridArr[row][colRow[1]-1].isShake&&!this.tileMap.gridArr[row][colRow[1]-1].isDown&&!this.tileMap.gridArr[row][colRow[1]-1].isUp){playEffect(22);this.tileMap.gridArr[row][colRow[1]-1].initDieEffectsType(6);if(this.tileMap.gridArr[row][colRow[1]-1].effectsType==1){this.tileMap.gridArr[row][colRow[1]-1].flameMonsterDie();}else{this.tileMap.gridArr[row][colRow[1]-1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1]-1]);}}

if(colRow[1]-1>=0&&colRow[1]-1<Global.classicModeWidth&&row+1>=3&&row+1<Global.classicModeHeight-1&&this.tileMap.gridArr[row+1][colRow[1]-1]!=null){if(!this.tileMap.gridArr[row+1][colRow[1]-1].isWaiteDie&&!this.tileMap.gridArr[row+1][colRow[1]-1].isShake&&!this.tileMap.gridArr[row+1][colRow[1]-1].isDown&&!this.tileMap.gridArr[row+1][colRow[1]-1].isUp){playEffect(22);this.tileMap.gridArr[row+1][colRow[1]-1].initDieEffectsType(6);if(this.tileMap.gridArr[row+1][colRow[1]-1].effectsType==1){this.tileMap.gridArr[row+1][colRow[1]-1].flameMonsterDie();}else{this.tileMap.gridArr[row+1][colRow[1]-1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row+1][colRow[1]-1]);}}

if(colRow[1]+1>=0&&colRow[1]+1<Global.classicModeWidth&&this.tileMap.gridArr[row][colRow[1]+1]!=null){if(!this.tileMap.gridArr[row][colRow[1]+1].isWaiteDie&&!this.tileMap.gridArr[row][colRow[1]+1].isShake&&!this.tileMap.gridArr[row][colRow[1]+1].isDown&&!this.tileMap.gridArr[row][colRow[1]+1].isUp){playEffect(22);this.tileMap.gridArr[row][colRow[1]+1].initDieEffectsType(6);if(this.tileMap.gridArr[row][colRow[1]+1].effectsType==1){this.tileMap.gridArr[row][colRow[1]+1].flameMonsterDie();}else{this.tileMap.gridArr[row][colRow[1]+1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row][colRow[1]+1]);}}

if(colRow[1]+1>=0&&colRow[1]+1<Global.classicModeWidth&&row+1>=3&&row+1<Global.classicModeHeight-1&&this.tileMap.gridArr[row+1][colRow[1]+1]!=null){if(!this.tileMap.gridArr[row+1][colRow[1]+1].isWaiteDie&&!this.tileMap.gridArr[row+1][colRow[1]+1].isShake&&!this.tileMap.gridArr[row+1][colRow[1]+1].isDown&&!this.tileMap.gridArr[row+1][colRow[1]+1].isUp){playEffect(22);this.tileMap.gridArr[row+1][colRow[1]+1].initDieEffectsType(6);if(this.tileMap.gridArr[row+1][colRow[1]+1].effectsType==1){this.tileMap.gridArr[row+1][colRow[1]+1].flameMonsterDie();}else{this.tileMap.gridArr[row+1][colRow[1]+1].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[row+1][colRow[1]+1]);}}}},lightningMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;for(var i=0;i<Global.classicModeWidth;i++){if(i!=colRow[1]&&this.tileMap.gridArr[colRow[0]][i]!=null){if(!this.tileMap.gridArr[colRow[0]][i].isWaiteDie&&!this.tileMap.gridArr[colRow[0]][i].isShake&&!this.tileMap.gridArr[colRow[0]][i].isDown&&!this.tileMap.gridArr[colRow[0]][i].isUp){playEffect(22);this.tileMap.gridArr[colRow[0]][i].initDieEffectsType(5);if(this.tileMap.gridArr[colRow[0]][i].effectsType==1){this.tileMap.gridArr[colRow[0]][i].flameMonsterDie();}else{this.tileMap.gridArr[colRow[0]][i].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[colRow[0]][i]);}}}

for(var j=0;j<9;j++){if(j+3!=colRow[0]&&this.tileMap.gridArr[j+3][colRow[1]]!=null){if(!this.tileMap.gridArr[j+3][colRow[1]].isWaiteDie&&!this.tileMap.gridArr[j+3][colRow[1]].isShake&&!this.tileMap.gridArr[j+3][colRow[1]].isDown&&!this.tileMap.gridArr[j+3][colRow[1]].isUp){playEffect(22);this.tileMap.gridArr[j+3][colRow[1]].initDieEffectsType(5);if(this.tileMap.gridArr[j+3][colRow[1]].effectsType==1){this.tileMap.gridArr[j+3][colRow[1]].flameMonsterDie();}else{this.tileMap.gridArr[j+3][colRow[1]].playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(this.tileMap.gridArr[j+3][colRow[1]]);}}}},blackHoleMonsterBoom:function(monster){if(monster==null)return;var colRow=monster.rowcol;var self=this;for(var j=colRow[0]-2;j<colRow[0]+3;j++){for(var i=colRow[1]-2;i<colRow[1]+3;i++){if(j>=3&&j<Global.classicModeHeight-2&&i>=0&&i<Global.classicModeWidth&&!(i==colRow[0]&&j==colRow[1])){var rotaMonster=this.tileMap.gridArr[j][i];if(rotaMonster!=null&&!rotaMonster.isWaiteDie&&!rotaMonster.isShake&&!rotaMonster.isDown&&!rotaMonster.isUp){cc.log("//播放吸入动画");rotaMonster.isWaiteDie=true;var pStart=rotaMonster.getPosition();var pEnd=monster.getPosition();rotaMonster.setMonsterAngle(pStart,pEnd);rotaMonster.addScheduleTime();rotaMonster.setMonsterData(pStart,pEnd);}}}}},useDyeingBottle:function(grid){var skin=randomNums(0,5);for(var i=-1;i<2;i++){for(var j=-1;j<2;j++){if(grid[0]+j>=0&&grid[0]+j<Global.classicModeWidth&&grid[1]+i>=3&&grid[1]+i<Global.classicModeHeight-1&&this.tileMap.gridArr[grid[1]+i][grid[0]+j]!=null){var monster=this.tileMap.gridArr[grid[1]+i][grid[0]+j];if(!monster.isShake&&(monster.isMonster||monster.isCageMonster)){monster.monster.setSkin(Global.monsterSkin[skin]);monster.skin=skin;}}}}

gamePlayerData.money=gamePlayerData.money-500;currentScene.gameLayer.refreshCurrentMoney();currentScene.gameLayer.isDyeing=false;currentScene.gameLayer.isSelectDyeBottle=false;currentScene.gameLayer.rootNode.dyebottlex\_1.visible=false;this.fallMonstersCallFun();},useToiletPlug:function(grid){cc.log("useToiletPlug");if(this.tileMap.gridArr[grid[1]][grid[0]]!=null){var skin=this.tileMap.gridArr[grid[1]][grid[0]].skin;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null){if(monster.skin!=null&&monster.skin==skin){monster.initDieEffectsType(1);if(monster.effectsType==1){monster.flameMonsterDie();}else{monster.playDieAni();}

this.gameControl.createRemoveSpecialMonsterList(monster);}}}}}

gamePlayerData.money=gamePlayerData.money-200;currentScene.gameLayer.refreshCurrentMoney();currentScene.gameLayer.isPlayToilet=false;currentScene.gameLayer.isSelectToiletSucker=false;currentScene.gameLayer.rootNode.toiletsuckerx\_2.visible=false;},removedFromBox:function(monster,type,pos){monster.monster.removeFromParent();monster.monster=null;var ID=monster.skin+1;monster.skin=-1;if(type==1){monster.isBoxMonster=false;monster.isBox=true;}

if(type==2){monster.isAdvancedBoxMonster=false;monster.isAdvancedBox=true;}

var tempMonster=new SpecialMonsterNode(ID);if(tempMonster!=null){tempMonster.x=pos.x;tempMonster.y=pos.y;this.touchPanel.addChild(tempMonster);}

return tempMonster;},updateCheckout:function(dt){this.checkShowGameAlarm();if(!this.isGameOver&&this.checkIsFullMonster()){this.checkGameOver();}},checkIsFullMonster:function(){var monsterCount=0;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null&&!monster.isShake){monsterCount++;}}}

if(monsterCount==63){return true;}else{return false;}},refreshTime:function(second,minute){var str=minute+":"+second;this.rootNode.BitmapFontLabel\_6.setString(str);},checkGameOver:function(){var data=new Object();data.lv=this.LV;var time=new Date();data.date=time.getFullYear()+"."+time.getMonth()+"."+time.getDate();data.score=this.score;gamePlayerData.gameRankingListData.push(data);sortGameRankingListData();if(this.score>=gamePlayerData.bestScore){gamePlayerData.bestScore=this.score;}

Global.gameOverData=[];Global.gameOverData.push(this.LV);Global.gameOverData.push(this.score);Global.gameOverData.push(gamePlayerData.bestScore);cc.log("弹游戏结束界面");this.isGameOver=true;var callFunAction=cc.callFunc(function(){currentScene.gameOver();});this.runAction(cc.sequence(cc.delayTime(1),callFunAction));savePlayerData();},calculateGetStar:function(){if(this.countTime>=this.gameStarThree){return 3;}else if(this.countTime>=this.gameStarTwo&&this.countTime<this.gameStarThree){return 2;}else if(this.countTime>=this.gameStarOne&&this.countTime<this.gameStarTwo){return 1;}else{cc.log("没有得星");return 0;}},showGameAlarm:function(boolean){if(boolean){this.rootNode.GameAlarm\_1.visible=true;this.rootNode.GameAlarm\_1.runAction(cc.sequence(cc.fadeIn(0.2),cc.fadeOut(0.5)).repeatForever());}else{this.rootNode.GameAlarm\_1.visible=false;}},checkShowGameAlarm:function(){var monsterCount=0;for(var i=3;i<Global.classicModeHeight-1;i++){for(var j=0;j<Global.classicModeWidth;j++){var monster=this.tileMap.gridArr[i][j];if(monster!=null&&!monster.isShake){monsterCount++;}}}

if(monsterCount>45){if(!this.isShowGameAlarm){this.isShowGameAlarm=true;this.showGameAlarm(this.isShowGameAlarm);this.unschedule(this.fallMonster);this.schedule(this.fallMonster,3);}}else{if(this.isShowGameAlarm){this.unschedule(this.fallMonster);this.schedule(this.fallMonster,10);}

this.isShowGameAlarm=false;this.showGameAlarm(this.isShowGameAlarm);}},showGameLevelUpNode:function(){this.gameLevelUpNode.BitmapFontLabel\_1.setString(""+this.LV);this.gameLevelUpNode.runAction(cc.sequence(cc.moveTo(1,cc.p(360,640)),cc.delayTime(0.8),cc.moveTo(0.5,cc.p(360,1920))));}});var WinLayer=cc.Layer.extend({rootNode:null,home:null,homePic:null,replay:null,replayPic:null,continues:null,continuesPic:null,rainPanel:null,gamewin:null,getScore:0,getStar:0,getGold:0,ctor:function(score,star){this.\_super();this.init(score,star);},init:function(score,star){this.initUI();var self=this;var callFunAction=cc.callFunc(function(){self.initText(score,star);self.initListener();});this.runAction(cc.sequence(cc.delayTime(1),callFunAction));},initUI:function(){this.rootNode=loadCSS(res.GameWinLayer\_json);this.addChild(this.rootNode);this.rootNode.BitmapFontLabel\_1.setString("");this.rootNode.BitmapFontLabel\_2.setString("");this.rootNode.BitmapFontLabel\_1.visible=false;this.rootNode.BitmapFontLabel\_2.visible=false;var animationPanel=this.rootNode.Panel\_2;cc.spriteFrameCache.addSpriteFrames(res.shengli\_plist);cc.spriteFrameCache.addSpriteFrames(res.shengli1\_plist);multipleAnimation(res.shengli\_json,res.shengli\_json);this.gamewin=new GameSkeletalAnimation();var data=skeletalAnimation[res.shengli\_json+"|shengli"];data.playCount=1;this.gamewin.initialize(data);this.gamewin.setFrameIntervalTime(24);this.gamewin.isRemoveFromComplete=false;animationPanel.addChild(this.gamewin);this.gamewin.x=380;this.gamewin.y=720;this.home=new GameSkeletalAnimation();var dataHome=skeletalAnimation[res.shengli\_json+"|gamehome"];dataHome.playCount=1;this.home.initialize(dataHome);this.home.setFrameIntervalTime(24);this.home.isRemoveFromComplete=false;animationPanel.addChild(this.home);this.home.x=200;this.home.y=510;this.replay=new GameSkeletalAnimation();var dataReplay=skeletalAnimation[res.shengli\_json+"|gamereplay"];dataReplay.playCount=1;this.replay.initialize(dataReplay);this.replay.setFrameIntervalTime(24);this.replay.isRemoveFromComplete=false;animationPanel.addChild(this.replay);this.replay.x=340;this.replay.y=450;this.continues=new GameSkeletalAnimation();var dataContinues=skeletalAnimation[res.shengli\_json+"|gamecontinues"];dataContinues.playCount=1;this.continues.initialize(dataContinues);this.continues.setFrameIntervalTime(24);this.continues.isRemoveFromComplete=false;animationPanel.addChild(this.continues);this.continues.x=500;this.continues.y=510;this.homePic=this.home.getBimapByUrl("#shenglipng17.png");this.replayPic=this.replay.getBimapByUrl("#shenglipng16.png");this.continuesPic=this.continues.getBimapByUrl("#shenglipng15.png");playEffect(6);},initText:function(score,star){this.getScore=score;this.getStar=star;this.getGold=star\*5;this.rootNode.BitmapFontLabel\_1.setString(score+"");this.rootNode.BitmapFontLabel\_2.setString(this.getGold+"");this.rootNode.BitmapFontLabel\_1.visible=true;this.rootNode.BitmapFontLabel\_2.visible=true;gamePlayerData.money+=this.getGold;currentScene.gameLayer.refreshCurrentMoney();savePlayerData();},initListener:function(){engine.gameEvent.createMouseDown(this.homePic,this.btnHomeFun);engine.gameEvent.createMouseDown(this.replayPic,this.btnReplayFun);engine.gameEvent.createMouseDown(this.continuesPic,this.btnContinuesFun);},showElement:function(){},btnHomeFun:function(){playEffect(2);cc.log("btnHomeFun");createCCSButtonScaling(this);transitions(SelectScene);},btnReplayFun:function(){playEffect(2);cc.log("btnReplayFun");createCCSButtonScaling(this);if(Global.currentSelectLevel==100){cc.log("Global.currentSelectLevel == 100");}else{Global.currentSelectLevel=Global.currentSelectLevel-1;}

transitions(GamePlayScene);},btnContinuesFun:function(){playEffect(2);cc.log("btnContinuesFun");createCCSButtonScaling(this);if(gamePlayerData.isJiHuo==0){if(Global.currentSelectLevel==Global.jieshuoPage){createSDKLayerFun();}else{transitions(GamePlayScene);}}else{transitions(GamePlayScene);}

savePlayerData();}});var DailyTaskLayer=cc.Layer.extend({rootNode:null,btnClose:null,eliminate1:null,button1:null,eliminate2:null,button2:null,eliminate3:null,button3:null,onEnter:function(){this.\_super();this.init();},init:function(){this.initUI();this.initText();this.initListener();},initText:function(){},initUI:function(){this.rootNode=loadCSS(res.dailyTaskLayer\_json);this.addChild(this.rootNode);this.btnClose=this.rootNode.Button\_1;this.eliminate1=this.rootNode.Eliminate;this.eliminate1.Sprite\_6.initWithFile(Global.dailyTaskMonster[gamePlayerData.gameRemoveMonster[0]]);this.eliminate1.BitmapFontLabel\_1.setString(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[0]]+"/"+gamePlayerData.gameRemoveMonster[1]);this.button1=this.eliminate1.Button\_2;this.button1.tag=1;this.button1.setPressedActionEnabled(true);this.eliminate2=this.rootNode.Eliminate\_0;this.eliminate2.Sprite\_6.initWithFile(Global.dailyTaskMonster[gamePlayerData.gameRemoveMonster[2]]);this.eliminate2.BitmapFontLabel\_1\_0.setString(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[2]]+"/"+gamePlayerData.gameRemoveMonster[3]);this.eliminate2.Image\_10.visible=false;this.button2=this.eliminate2.Button\_3;this.button2.tag=2;this.button2.setPressedActionEnabled(true);this.eliminate3=this.rootNode.Eliminate\_1;this.eliminate3.BitmapFontLabel\_1\_0.setString(""+gamePlayerData.gameRemoveMonster[4]);this.eliminate3.Image\_10.visible=false;this.button3=this.eliminate3.Button\_6;this.button3.tag=3;this.button3.setPressedActionEnabled(true);if(gamePlayerData.isCompleteDailyTsk[0]){this.eliminate1.Image\_10.visible=true;this.button1.visible=false;}else{this.eliminate1.Image\_10.visible=false;this.button1.visible=true;}

if(gamePlayerData.isCompleteDailyTsk[1]){this.eliminate2.Image\_10.visible=true;this.button2.visible=false;}else{this.eliminate2.Image\_10.visible=false;this.button2.visible=true;}

if(gamePlayerData.isCompleteDailyTsk[2]){this.eliminate3.Image\_10.visible=true;this.button3.visible=false;}else{this.eliminate3.Image\_10.visible=false;this.button3.visible=true;}

savePlayerData();},initListener:function(){this.btnClose.addTouchEventListener(this.btnCloseFun);this.button1.addTouchEventListener(this.btnTrueFun);this.button2.addTouchEventListener(this.btnTrueFun);this.button3.addTouchEventListener(this.btnTrueFun);},btnCloseFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnCloseFun");if(currentScene.gameTaskLayer!=null){currentScene.gameTaskLayer.removeFromParent();currentScene.gameTaskLayer=null;}

break;}},btnTrueFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:cc.log("btnTrueFun"+sender.tag);switch(sender.tag){case 1:if(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[0]]>=gamePlayerData.gameRemoveMonster[1]){playEffect(26);gamePlayerData.money+=50;currentScene.gameTaskLayer.button1.visible=false;currentScene.gameTaskLayer.eliminate1.Image\_10.visible=true;gamePlayerData.isCompleteDailyTsk[0]=true;}

break;case 2:if(gamePlayerData.gameRecordRemoveMonster[gamePlayerData.gameRemoveMonster[2]]>=gamePlayerData.gameRemoveMonster[3]){playEffect(26);gamePlayerData.money+=50;currentScene.gameTaskLayer.button2.visible=false;currentScene.gameTaskLayer.eliminate2.Image\_10.visible=true;gamePlayerData.isCompleteDailyTsk[1]=true;}

break;case 3:if(gamePlayerData.currentLevel>=gamePlayerData.gameRemoveMonster[4]){playEffect(26);gamePlayerData.money+=50;currentScene.gameTaskLayer.button3.visible=false;currentScene.gameTaskLayer.eliminate3.Image\_10.visible=true;gamePlayerData.isCompleteDailyTsk[2]=true;}

break;}

break;var RankingListLayer=cc.Layer.extend({rootNode:null,btnBack:null,scrollPanel:null,onEnter:function(){this.\_super();this.init();},init:function(){this.initUI();this.initText();this.initItem();this.initListener();},initText:function(){},initUI:function(){this.rootNode=loadCSS(res.RankingListLayer\_json);this.addChild(this.rootNode);this.btnBack=this.rootNode.Button\_1;this.scrollPanel=this.rootNode.Image\_1.ScrollView\_1;},initListener:function(){this.btnBack.addTouchEventListener(this.btnBackFun);},initItem:function(){for(var i=0;i<10;i++){var itemNode=ccs.load(res.RankingListNode\_json).node;itemNode.x=280;itemNode.y=1500-i\*160;this.scrollPanel.addChild(itemNode);if(i!=0){itemNode.getChildByName("Image\_2").getChildByName("Image\_3").visible=false;}

var sp=cc.spriteFrameCache.getSpriteFrame("rank"+(i+1)+".png");itemNode.getChildByName("Image\_2").getChildByName("Sprite\_2").setSpriteFrame(sp);if(gamePlayerData.gameRankingListData[i]!=null){itemNode.getChildByName("BitmapFontLabel\_1").setString(""+gamePlayerData.gameRankingListData[i].lv);itemNode.getChildByName("BitmapFontLabel\_2").setString(""+gamePlayerData.gameRankingListData[i].date);itemNode.getChildByName("BitmapFontLabel\_3").setString(""+gamePlayerData.gameRankingListData[i].score);}else{itemNode.getChildByName("BitmapFontLabel\_1").visible=false;itemNode.getChildByName("BitmapFontLabel\_2").visible=false;itemNode.getChildByName("BitmapFontLabel\_3").visible=false;}}},btnBackFun:function(sender,type){switch(type){case ccui.Widget.TOUCH\_ENDED:playEffect(2);cc.log("btnBackFun");if(currentScene.gameRankLayer!=null){currentScene.gameRankLayer.removeFromParent();currentScene.gameRankLayer=null;}

break;}}});}

savePlayerData();}});ccui.HBox=ccui.Layout.extend({ctor:function(size){ccui.Layout.prototype.ctor.call(this);this.setLayoutType(ccui.Layout.LINEAR\_HORIZONTAL);if(size){this.setContentSize(size);}}});ccui.HBox.create=function(size){return new ccui.HBox(size);};(function(){if(cc===undefined){return;}

var PluginManager=function(){};PluginManager.prototype={constructor:PluginManager,getInstance:function(){return this;},loadPlugin:function(pluginName){},unloadPlugin:function(pluginName){}};var PluginAssembly=function(){};PluginAssembly.prototype={constructor:PluginAssembly,setDebugMode:function(debug){},startSession:function(appKey){},setCaptureUncaughtException:function(Capture){},callFuncWithParam:function(funName){if(typeof this[funName]==='function'){return this[funName].apply(this,Array.prototype.splice.call(arguments,1));}else{cc.log("function is not define");}},callStringFuncWithParam:function(funName){this.callFuncWithParam.apply(arguments);},getPluginName:function(){return this.\_name;},getPluginVersion:function(){return this.\_version;}};PluginAssembly.extend=function(name,porp){var p,prototype={};for(p in PluginAssembly.prototype){prototype[p]=PluginAssembly.prototype[p];}

for(p in porp){prototype[p]=porp[p];}

var tmp=eval("(function "+name+"Plugin(){})");prototype.constructor=tmp;tmp.prototype=prototype;return tmp;};var Param=function(type,value){var paramType=plugin.PluginParam.ParamType,tmpValue;switch(type){case paramType.TypeInt:tmpValue=parseInt(value);break;case paramType.TypeFloat:tmpValue=parseFloat(value);break;case paramType.TypeBool:tmpValue=Boolean(value);break;case paramType.TypeString:tmpValue=String(value);break;case paramType.TypeStringMap:tmpValue=value

break;default:tmpValue=value;}

return tmpValue};Param.ParamType={TypeInt:1,TypeFloat:2,TypeBool:3,TypeString:4,TypeStringMap:5};Param.AdsResultCode={AdsReceived:0,FullScreenViewShown:1,FullScreenViewDismissed:2,PointsSpendSucceed:3,PointsSpendFailed:4,NetworkError:5,UnknownError:6};Param.PayResultCode={PaySuccess:0,PayFail:1,PayCancel:2,PayTimeOut:3};Param.ShareResultCode={ShareSuccess:0,ShareFail:1,ShareCancel:2,ShareTimeOut:3};var PluginList={};var Plugin={extend:function(name,extend){var config=(cc.game.config&&cc.game.config.plugin)||{};PluginList[name]=new(PluginAssembly.extend(name,extend));typeof PluginList[name].ctor==="function"&&PluginList[name].ctor(config[name]);},PluginList:PluginList,PluginParam:Param,PluginManager:new PluginManager()};window.plugin=Plugin;})();ccui.Layout=ccui.Widget.extend({\_clippingEnabled:false,\_backGroundScale9Enabled:null,\_backGroundImage:null,\_backGroundImageFileName:null,\_backGroundImageCapInsets:null,\_colorType:null,\_bgImageTexType:ccui.Widget.LOCAL\_TEXTURE,\_colorRender:null,\_gradientRender:null,\_color:null,\_startColor:null,\_endColor:null,\_alongVector:null,\_opacity:255,\_backGroundImageTextureSize:null,\_layoutType:null,\_doLayoutDirty:true,\_clippingRectDirty:true,\_clippingType:null,\_clippingStencil:null,\_scissorRectDirty:false,\_clippingRect:null,\_clippingParent:null,\_className:"Layout",\_backGroundImageColor:null,\_finalPositionX:0,\_finalPositionY:0,\_backGroundImageOpacity:0,\_loopFocus:false,\_\_passFocusToChild:true,\_isFocusPassing:false,\_isInterceptTouch:false,ctor:function(){this.\_layoutType=ccui.Layout.ABSOLUTE;this.\_widgetType=ccui.Widget.TYPE\_CONTAINER;this.\_clippingType=ccui.Layout.CLIPPING\_SCISSOR;this.\_colorType=ccui.Layout.BG\_COLOR\_NONE;ccui.Widget.prototype.ctor.call(this);this.ignoreContentAdaptWithSize(false);this.setContentSize(cc.size(0,0));this.setAnchorPoint(0,0);this.onPassFocusToChild=this.\_findNearestChildWidgetIndex.bind(this);this.\_backGroundImageCapInsets=cc.rect(0,0,0,0);this.\_color=cc.color(255,255,255,255);this.\_startColor=cc.color(255,255,255,255);this.\_endColor=cc.color(255,255,255,255);this.\_alongVector=cc.p(0,-1);this.\_backGroundImageTextureSize=cc.size(0,0);this.\_clippingRect=cc.rect(0,0,0,0);this.\_backGroundImageColor=cc.color(255,255,255,255);},onEnter:function(){ccui.Widget.prototype.onEnter.call(this);if(this.\_clippingStencil)this.\_clippingStencil.onEnter();this.\_doLayoutDirty=true;this.\_clippingRectDirty=true;},onExit:function(){ccui.Widget.prototype.onExit.call(this);if(this.\_clippingStencil)this.\_clippingStencil.onExit();},setLoopFocus:function(loop){this.\_loopFocus=loop;},isLoopFocus:function(){return this.\_loopFocus;},setPassFocusToChild:function(pass){this.\_\_passFocusToChild=pass;},isPassFocusToChild:function(){return this.\_\_passFocusToChild;},findNextFocusedWidget:function(direction,current){if(this.\_isFocusPassing||this.isFocused()){var parent=this.getParent();this.\_isFocusPassing=false;if(this.\_\_passFocusToChild){var w=this.\_passFocusToChild(direction,current);if(w instanceof ccui.Layout&&parent){parent.\_isFocusPassing=true;return parent.findNextFocusedWidget(direction,this);}

return w;}

if(null==parent||!(parent instanceof ccui.Layout))return this;parent.\_isFocusPassing=true;return parent.findNextFocusedWidget(direction,this);}else if(current.isFocused()||current instanceof ccui.Layout){if(this.\_layoutType===ccui.Layout.LINEAR\_HORIZONTAL){switch(direction){case ccui.Widget.LEFT:return this.\_getPreviousFocusedWidget(direction,current);break;case ccui.Widget.RIGHT:return this.\_getNextFocusedWidget(direction,current);break;case ccui.Widget.DOWN:case ccui.Widget.UP:if(this.\_isLastWidgetInContainer(this,direction)){if(this.\_isWidgetAncestorSupportLoopFocus(current,direction))return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);return current;}else{return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);}

break;default:cc.assert(0,"Invalid Focus Direction");return current;}}else if(this.\_layoutType===ccui.Layout.LINEAR\_VERTICAL){switch(direction){case ccui.Widget.LEFT:case ccui.Widget.RIGHT:if(this.\_isLastWidgetInContainer(this,direction)){if(this.\_isWidgetAncestorSupportLoopFocus(current,direction))return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);return current;}else return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);break;case ccui.Widget.DOWN:return this.\_getNextFocusedWidget(direction,current);break;case ccui.Widget.UP:return this.\_getPreviousFocusedWidget(direction,current);break;default:cc.assert(0,"Invalid Focus Direction");return current;}}else{cc.assert(0,"Un Supported Layout type, please use VBox and HBox instead!!!");return current;}}else return current;},onPassFocusToChild:null,addChild:function(widget,zOrder,tag){if((widget instanceof ccui.Widget)){this.\_supplyTheLayoutParameterLackToChild(widget);}

ccui.Widget.prototype.addChild.call(this,widget,zOrder,tag);this.\_doLayoutDirty=true;},removeChild:function(widget,cleanup){ccui.Widget.prototype.removeChild.call(this,widget,cleanup);this.\_doLayoutDirty=true;},removeAllChildren:function(cleanup){ccui.Widget.prototype.removeAllChildren.call(this,cleanup);this.\_doLayoutDirty=true;},removeAllChildrenWithCleanup:function(cleanup){ccui.Widget.prototype.removeAllChildrenWithCleanup.call(this,cleanup);this.\_doLayoutDirty=true;},isClippingEnabled:function(){return this.\_clippingEnabled;},visit:function(parentCmd){if(!this.\_visible)return;this.\_adaptRenderers();this.\_doLayout();if(this.\_clippingEnabled){switch(this.\_clippingType){case ccui.Layout.CLIPPING\_STENCIL:this.\_renderCmd.stencilClippingVisit(parentCmd);break;case ccui.Layout.CLIPPING\_SCISSOR:this.\_renderCmd.scissorClippingVisit(parentCmd);break;default:break;}}else{ccui.Widget.prototype.visit.call(this,parentCmd);}},setClippingEnabled:function(able){if(able===this.\_clippingEnabled)return;this.\_clippingEnabled=able;switch(this.\_clippingType){case ccui.Layout.CLIPPING\_SCISSOR:case ccui.Layout.CLIPPING\_STENCIL:if(able){this.\_clippingStencil=new cc.DrawNode();this.\_renderCmd.rebindStencilRendering(this.\_clippingStencil);if(this.\_running)this.\_clippingStencil.onEnter();this.\_setStencilClippingSize(this.\_contentSize);}else{if(this.\_running&&this.\_clippingStencil)this.\_clippingStencil.onExit();this.\_clippingStencil=null;}

break;default:break;}},setClippingType:function(type){if(type===this.\_clippingType)return;var clippingEnabled=this.isClippingEnabled();this.setClippingEnabled(false);this.\_clippingType=type;this.setClippingEnabled(clippingEnabled);},getClippingType:function(){return this.\_clippingType;},\_setStencilClippingSize:function(size){if(this.\_clippingEnabled){var rect=[];rect[0]=cc.p(0,0);rect[1]=cc.p(size.width,0);rect[2]=cc.p(size.width,size.height);rect[3]=cc.p(0,size.height);var green=cc.color.GREEN;this.\_clippingStencil.clear();this.\_clippingStencil.setLocalBB&&this.\_clippingStencil.setLocalBB(0,0,size.width,size.height);this.\_clippingStencil.drawPoly(rect,4,green,0,green);}},\_getClippingRect:function(){if(this.\_clippingRectDirty){var worldPos=this.convertToWorldSpace(cc.p(0,0));var t=this.getNodeToWorldTransform();var scissorWidth=this.\_contentSize.width\*t.a;var scissorHeight=this.\_contentSize.height\*t.d;var parentClippingRect;var parent=this;while(parent){parent=parent.getParent();if(parent&&parent instanceof ccui.Layout&&parent.isClippingEnabled()){this.\_clippingParent=parent;break;}}

if(this.\_clippingParent){parentClippingRect=this.\_clippingParent.\_getClippingRect();this.\_clippingRect.x=Math.max(worldPos.x,parentClippingRect.x);this.\_clippingRect.y=Math.max(worldPos.y,parentClippingRect.y);var right=Math.min(worldPos.x+scissorWidth,parentClippingRect.x+parentClippingRect.width);var top=Math.min(worldPos.y+scissorHeight,parentClippingRect.y+parentClippingRect.height);this.\_clippingRect.width=Math.max(0.0,right-this.\_clippingRect.x);this.\_clippingRect.height=Math.max(0.0,top-this.\_clippingRect.y);}else{this.\_clippingRect.x=worldPos.x;this.\_clippingRect.y=worldPos.y;this.\_clippingRect.width=scissorWidth;this.\_clippingRect.height=scissorHeight;}

this.\_clippingRectDirty=false;}

return this.\_clippingRect;},\_onSizeChanged:function(){ccui.Widget.prototype.\_onSizeChanged.call(this);var locContentSize=this.\_contentSize;this.\_setStencilClippingSize(locContentSize);this.\_doLayoutDirty=true;this.\_clippingRectDirty=true;if(this.\_backGroundImage){this.\_backGroundImage.setPosition(locContentSize.width\*0.5,locContentSize.height\*0.5);if(this.\_backGroundScale9Enabled&&this.\_backGroundImage instanceof ccui.Scale9Sprite)this.\_backGroundImage.setPreferredSize(locContentSize);}

if(this.\_colorRender)this.\_colorRender.setContentSize(locContentSize);if(this.\_gradientRender)this.\_gradientRender.setContentSize(locContentSize);},setBackGroundImageScale9Enabled:function(able){if(this.\_backGroundScale9Enabled===able)return;this.removeProtectedChild(this.\_backGroundImage);this.\_backGroundImage=null;this.\_backGroundScale9Enabled=able;this.\_addBackGroundImage();this.setBackGroundImage(this.\_backGroundImageFileName,this.\_bgImageTexType);this.setBackGroundImageCapInsets(this.\_backGroundImageCapInsets);},isBackGroundImageScale9Enabled:function(){return this.\_backGroundScale9Enabled;},setBackGroundImage:function(fileName,texType){if(!fileName)return;texType=texType||ccui.Widget.LOCAL\_TEXTURE;if(this.\_backGroundImage===null){this.\_addBackGroundImage();this.setBackGroundImageScale9Enabled(this.\_backGroundScale9Enabled);}

this.\_backGroundImageFileName=fileName;this.\_bgImageTexType=texType;var locBackgroundImage=this.\_backGroundImage;switch(this.\_bgImageTexType){case ccui.Widget.LOCAL\_TEXTURE:locBackgroundImage.initWithFile(fileName);break;case ccui.Widget.PLIST\_TEXTURE:locBackgroundImage.initWithSpriteFrameName(fileName);break;default:break;}

if(this.\_backGroundScale9Enabled)locBackgroundImage.setPreferredSize(this.\_contentSize);this.\_backGroundImageTextureSize=locBackgroundImage.getContentSize();locBackgroundImage.setPosition(this.\_contentSize.width\*0.5,this.\_contentSize.height\*0.5);this.\_updateBackGroundImageColor();},setBackGroundImageCapInsets:function(capInsets){if(!capInsets)return;var locInsets=this.\_backGroundImageCapInsets;locInsets.x=capInsets.x;locInsets.y=capInsets.y;locInsets.width=capInsets.width;locInsets.height=capInsets.height;if(this.\_backGroundScale9Enabled)this.\_backGroundImage.setCapInsets(capInsets);},getBackGroundImageCapInsets:function(){return cc.rect(this.\_backGroundImageCapInsets);},\_supplyTheLayoutParameterLackToChild:function(locChild){if(!locChild){return;}

switch(this.\_layoutType){case ccui.Layout.ABSOLUTE:break;case ccui.Layout.LINEAR\_HORIZONTAL:case ccui.Layout.LINEAR\_VERTICAL:var layoutParameter=locChild.getLayoutParameter(ccui.LayoutParameter.LINEAR);if(!layoutParameter)locChild.setLayoutParameter(new ccui.LinearLayoutParameter());break;case ccui.Layout.RELATIVE:var layoutParameter=locChild.getLayoutParameter(ccui.LayoutParameter.RELATIVE);if(!layoutParameter)locChild.setLayoutParameter(new ccui.RelativeLayoutParameter());break;default:break;}},\_addBackGroundImage:function(){var contentSize=this.\_contentSize;if(this.\_backGroundScale9Enabled){this.\_backGroundImage=new ccui.Scale9Sprite();this.\_backGroundImage.setPreferredSize(contentSize);}else this.\_backGroundImage=new cc.Sprite();this.addProtectedChild(this.\_backGroundImage,ccui.Layout.BACKGROUND\_IMAGE\_ZORDER,-1);this.\_backGroundImage.setPosition(contentSize.width\*0.5,contentSize.height\*0.5);},removeBackGroundImage:function(){if(!this.\_backGroundImage)return;this.removeProtectedChild(this.\_backGroundImage);this.\_backGroundImage=null;this.\_backGroundImageFileName="";this.\_backGroundImageTextureSize.width=0;this.\_backGroundImageTextureSize.height=0;},setBackGroundColorType:function(type){if(this.\_colorType===type)return;switch(this.\_colorType){case ccui.Layout.BG\_COLOR\_NONE:if(this.\_colorRender){this.removeProtectedChild(this.\_colorRender);this.\_colorRender=null;}

if(this.\_gradientRender){this.removeProtectedChild(this.\_gradientRender);this.\_gradientRender=null;}

break;case ccui.Layout.BG\_COLOR\_SOLID:if(this.\_colorRender){this.removeProtectedChild(this.\_colorRender);this.\_colorRender=null;}

break;case ccui.Layout.BG\_COLOR\_GRADIENT:if(this.\_gradientRender){this.removeProtectedChild(this.\_gradientRender);this.\_gradientRender=null;}

break;default:break;}

this.\_colorType=type;switch(this.\_colorType){case ccui.Layout.BG\_COLOR\_NONE:break;case ccui.Layout.BG\_COLOR\_SOLID:this.\_colorRender=new cc.LayerColor();this.\_colorRender.setContentSize(this.\_contentSize);this.\_colorRender.setOpacity(this.\_opacity);this.\_colorRender.setColor(this.\_color);this.addProtectedChild(this.\_colorRender,ccui.Layout.BACKGROUND\_RENDERER\_ZORDER,-1);break;case ccui.Layout.BG\_COLOR\_GRADIENT:this.\_gradientRender=new cc.LayerGradient(cc.color(255,0,0,255),cc.color(0,255,0,255));this.\_gradientRender.setContentSize(this.\_contentSize);this.\_gradientRender.setOpacity(this.\_opacity);this.\_gradientRender.setStartColor(this.\_startColor);this.\_gradientRender.setEndColor(this.\_endColor);this.\_gradientRender.setVector(this.\_alongVector);this.addProtectedChild(this.\_gradientRender,ccui.Layout.BACKGROUND\_RENDERER\_ZORDER,-1);break;default:break;}},getBackGroundColorType:function(){return this.\_colorType;},setBackGroundColor:function(color,endColor){if(!endColor){this.\_color.r=color.r;this.\_color.g=color.g;this.\_color.b=color.b;if(this.\_colorRender)this.\_colorRender.setColor(color);}else{this.\_startColor.r=color.r;this.\_startColor.g=color.g;this.\_startColor.b=color.b;if(this.\_gradientRender)this.\_gradientRender.setStartColor(color);this.\_endColor.r=endColor.r;this.\_endColor.g=endColor.g;this.\_endColor.b=endColor.b;if(this.\_gradientRender)this.\_gradientRender.setEndColor(endColor);}},getBackGroundColor:function(){var tmpColor=this.\_color;return cc.color(tmpColor.r,tmpColor.g,tmpColor.b,tmpColor.a);},getBackGroundStartColor:function(){var tmpColor=this.\_startColor;return cc.color(tmpColor.r,tmpColor.g,tmpColor.b,tmpColor.a);},getBackGroundEndColor:function(){var tmpColor=this.\_endColor;return cc.color(tmpColor.r,tmpColor.g,tmpColor.b,tmpColor.a);},setBackGroundColorOpacity:function(opacity){this.\_opacity=opacity;switch(this.\_colorType){case ccui.Layout.BG\_COLOR\_NONE:break;case ccui.Layout.BG\_COLOR\_SOLID:this.\_colorRender.setOpacity(opacity);break;case ccui.Layout.BG\_COLOR\_GRADIENT:this.\_gradientRender.setOpacity(opacity);break;default:break;}},getBackGroundColorOpacity:function(){return this.\_opacity;},setBackGroundColorVector:function(vector){this.\_alongVector.x=vector.x;this.\_alongVector.y=vector.y;if(this.\_gradientRender){this.\_gradientRender.setVector(vector);}},getBackGroundColorVector:function(){return this.\_alongVector;},setBackGroundImageColor:function(color){this.\_backGroundImageColor.r=color.r;this.\_backGroundImageColor.g=color.g;this.\_backGroundImageColor.b=color.b;this.\_updateBackGroundImageColor();},setBackGroundImageOpacity:function(opacity){this.\_backGroundImageColor.a=opacity;this.getBackGroundImageColor();},getBackGroundImageColor:function(){var color=this.\_backGroundImageColor;return cc.color(color.r,color.g,color.b,color.a);},getBackGroundImageOpacity:function(){return this.\_backGroundImageColor.a;},\_updateBackGroundImageColor:function(){if(this.\_backGroundImage)this.\_backGroundImage.setColor(this.\_backGroundImageColor);},getBackGroundImageTextureSize:function(){return this.\_backGroundImageTextureSize;},setLayoutType:function(type){this.\_layoutType=type;var layoutChildrenArray=this.\_children;var locChild=null;for(var i=0;i<layoutChildrenArray.length;i++){locChild=layoutChildrenArray[i];if(locChild instanceof ccui.Widget)this.\_supplyTheLayoutParameterLackToChild(locChild);}

this.\_doLayoutDirty=true;},getLayoutType:function(){return this.\_layoutType;},requestDoLayout:function(){this.\_doLayoutDirty=true;},\_doLayout:function(){if(!this.\_doLayoutDirty)return;this.sortAllChildren();var executant=ccui.getLayoutManager(this.\_layoutType);if(executant)executant.\_doLayout(this);this.\_doLayoutDirty=false;},\_getLayoutContentSize:function(){return this.getContentSize();},\_getLayoutElements:function(){return this.getChildren();},\_updateBackGroundImageOpacity:function(){if(this.\_backGroundImage)this.\_backGroundImage.setOpacity(this.\_backGroundImageOpacity);},\_updateBackGroundImageRGBA:function(){if(this.\_backGroundImage){this.\_backGroundImage.setColor(this.\_backGroundImageColor);this.\_backGroundImage.setOpacity(this.\_backGroundImageOpacity);}},\_getLayoutAccumulatedSize:function(){var children=this.getChildren();var layoutSize=cc.size(0,0);var widgetCount=0,locSize;for(var i=0,len=children.length;i<len;i++){var layout=children[i];if(null!==layout&&layout instanceof ccui.Layout){locSize=layout.\_getLayoutAccumulatedSize();layoutSize.width+=locSize.width;layoutSize.height+=locSize.height;}else{if(layout instanceof ccui.Widget){widgetCount++;var m=layout.getLayoutParameter().getMargin();locSize=layout.getContentSize();layoutSize.width+=locSize.width+(m.right+m.left)\*0.5;layoutSize.height+=locSize.height+(m.top+m.bottom)\*0.5;}}}

var type=this.getLayoutType();if(type===ccui.Layout.LINEAR\_HORIZONTAL)layoutSize.height=layoutSize.height-layoutSize.height/widgetCount\*(widgetCount-1);if(type===ccui.Layout.LINEAR\_VERTICAL)layoutSize.width=layoutSize.width-layoutSize.width/widgetCount\*(widgetCount-1);return layoutSize;},\_findNearestChildWidgetIndex:function(direction,baseWidget){if(baseWidget==null||baseWidget===this)return this.\_findFirstFocusEnabledWidgetIndex();var index=0,locChildren=this.getChildren();var count=locChildren.length,widgetPosition;var distance=cc.FLT\_MAX,found=0;if(direction===ccui.Widget.LEFT||direction===ccui.Widget.RIGHT||direction===ccui.Widget.DOWN||direction===ccui.Widget.UP){widgetPosition=this.\_getWorldCenterPoint(baseWidget);while(index<count){var w=locChildren[index];if(w&&w instanceof ccui.Widget&&w.isFocusEnabled()){var length=(w instanceof ccui.Layout)?w.\_calculateNearestDistance(baseWidget):cc.pLength(cc.pSub(this.\_getWorldCenterPoint(w),widgetPosition));if(length<distance){found=index;distance=length;}}

index++;}

return found;}

cc.log("invalid focus direction!");return 0;},\_findFarthestChildWidgetIndex:function(direction,baseWidget){if(baseWidget==null||baseWidget===this)return this.\_findFirstFocusEnabledWidgetIndex();var index=0,locChildren=this.getChildren();var count=locChildren.length;var distance=-cc.FLT\_MAX,found=0;if(direction===ccui.Widget.LEFT||direction===ccui.Widget.RIGHT||direction===ccui.Widget.DOWN||direction===ccui.Widget.UP){var widgetPosition=this.\_getWorldCenterPoint(baseWidget);while(index<count){var w=locChildren[index];if(w&&w instanceof ccui.Widget&&w.isFocusEnabled()){var length=(w instanceof ccui.Layout)?w.\_calculateFarthestDistance(baseWidget):cc.pLength(cc.pSub(this.\_getWorldCenterPoint(w),widgetPosition));if(length>distance){found=index;distance=length;}}

index++;}

return found;}

cc.log("invalid focus direction!!!");return 0;},\_calculateNearestDistance:function(baseWidget){var distance=cc.FLT\_MAX;var widgetPosition=this.\_getWorldCenterPoint(baseWidget);var locChildren=this.\_children;for(var i=0,len=locChildren.length;i<len;i++){var widget=locChildren[i],length;if(widget instanceof ccui.Layout)length=widget.\_calculateNearestDistance(baseWidget);else{if(widget instanceof ccui.Widget&&widget.isFocusEnabled())length=cc.pLength(cc.pSub(this.\_getWorldCenterPoint(widget),widgetPosition));else continue;}

if(length<distance)distance=length;}

return distance;},\_calculateFarthestDistance:function(baseWidget){var distance=-cc.FLT\_MAX;var widgetPosition=this.\_getWorldCenterPoint(baseWidget);var locChildren=this.\_children;for(var i=0,len=locChildren.length;i<len;i++){var layout=locChildren[i];var length;if(layout instanceof ccui.Layout)length=layout.\_calculateFarthestDistance(baseWidget);else{if(layout instanceof ccui.Widget&&layout.isFocusEnabled()){var wPosition=this.\_getWorldCenterPoint(layout);length=cc.pLength(cc.pSub(wPosition,widgetPosition));}else continue;}

if(length>distance)distance=length;}

return distance;},\_findProperSearchingFunctor:function(direction,baseWidget){if(baseWidget===undefined)return;var previousWidgetPosition=this.\_getWorldCenterPoint(baseWidget);var widgetPosition=this.\_getWorldCenterPoint(this.\_findFirstNonLayoutWidget());if(direction===ccui.Widget.LEFT){this.onPassFocusToChild=(previousWidgetPosition.x>widgetPosition.x)?this.\_findNearestChildWidgetIndex:this.\_findFarthestChildWidgetIndex;}else if(direction===ccui.Widget.RIGHT){this.onPassFocusToChild=(previousWidgetPosition.x>widgetPosition.x)?this.\_findFarthestChildWidgetIndex:this.\_findNearestChildWidgetIndex;}else if(direction===ccui.Widget.DOWN){this.onPassFocusToChild=(previousWidgetPosition.y>widgetPosition.y)?this.\_findNearestChildWidgetIndex:this.\_findFarthestChildWidgetIndex;}else if(direction===ccui.Widget.UP){this.onPassFocusToChild=(previousWidgetPosition.y<widgetPosition.y)?this.\_findNearestChildWidgetIndex:this.\_findFarthestChildWidgetIndex;}else cc.log("invalid direction!");},\_findFirstNonLayoutWidget:function(){var locChildren=this.\_children;for(var i=0,len=locChildren.length;i<len;i++){var child=locChildren[i];if(child instanceof ccui.Layout){var widget=child.\_findFirstNonLayoutWidget();if(widget)return widget;}else{if(child instanceof ccui.Widget)return child;}}

return null;},\_findFirstFocusEnabledWidgetIndex:function(){var index=0,locChildren=this.getChildren();var count=locChildren.length;while(index<count){var w=locChildren[index];if(w&&w instanceof ccui.Widget&&w.isFocusEnabled())return index;index++;}

return 0;},\_findFocusEnabledChildWidgetByIndex:function(index){var widget=this.\_getChildWidgetByIndex(index);if(widget){if(widget.isFocusEnabled())return widget;index=index+1;return this.\_findFocusEnabledChildWidgetByIndex(index);}

return null;},\_getWorldCenterPoint:function(widget){var widgetSize=widget instanceof ccui.Layout?widget.\_getLayoutAccumulatedSize():widget.getContentSize();return widget.convertToWorldSpace(cc.p(widgetSize.width/2,widgetSize.height/2));},\_getNextFocusedWidget:function(direction,current){var nextWidget=null,locChildren=this.\_children;var previousWidgetPos=locChildren.indexOf(current);previousWidgetPos=previousWidgetPos+1;if(previousWidgetPos<locChildren.length){nextWidget=this.\_getChildWidgetByIndex(previousWidgetPos);if(nextWidget){if(nextWidget.isFocusEnabled()){if(nextWidget instanceof ccui.Layout){nextWidget.\_isFocusPassing=true;return nextWidget.findNextFocusedWidget(direction,nextWidget);}else{this.dispatchFocusEvent(current,nextWidget);return nextWidget;}}else return this.\_getNextFocusedWidget(direction,nextWidget);}else return current;}else{if(this.\_loopFocus){if(this.\_checkFocusEnabledChild()){previousWidgetPos=0;nextWidget=this.\_getChildWidgetByIndex(previousWidgetPos);if(nextWidget.isFocusEnabled()){if(nextWidget instanceof ccui.Layout){nextWidget.\_isFocusPassing=true;return nextWidget.findNextFocusedWidget(direction,nextWidget);}else{this.dispatchFocusEvent(current,nextWidget);return nextWidget;}}else return this.\_getNextFocusedWidget(direction,nextWidget);}else return(current instanceof ccui.Layout)?current:ccui.Widget.\_focusedWidget;}else{if(this.\_isLastWidgetInContainer(current,direction)){if(this.\_isWidgetAncestorSupportLoopFocus(this,direction))return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);return(current instanceof ccui.Layout)?current:ccui.Widget.\_focusedWidget;}else return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);}}},\_getPreviousFocusedWidget:function(direction,current){var nextWidget=null,locChildren=this.\_children;var previousWidgetPos=locChildren.indexOf(current);previousWidgetPos=previousWidgetPos-1;if(previousWidgetPos>=0){nextWidget=this.\_getChildWidgetByIndex(previousWidgetPos);if(nextWidget.isFocusEnabled()){if(nextWidget instanceof ccui.Layout){nextWidget.\_isFocusPassing=true;return nextWidget.findNextFocusedWidget(direction,nextWidget);}

this.dispatchFocusEvent(current,nextWidget);return nextWidget;}else return this.\_getPreviousFocusedWidget(direction,nextWidget);}else{if(this.\_loopFocus){if(this.\_checkFocusEnabledChild()){previousWidgetPos=locChildren.length-1;nextWidget=this.\_getChildWidgetByIndex(previousWidgetPos);if(nextWidget.isFocusEnabled()){if(nextWidget instanceof ccui.Layout){nextWidget.\_isFocusPassing=true;return nextWidget.findNextFocusedWidget(direction,nextWidget);}else{this.dispatchFocusEvent(current,nextWidget);return nextWidget;}}else return this.\_getPreviousFocusedWidget(direction,nextWidget);}else return(current instanceof ccui.Layout)?current:ccui.Widget.\_focusedWidget;}else{if(this.\_isLastWidgetInContainer(current,direction)){if(this.\_isWidgetAncestorSupportLoopFocus(this,direction))return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);return(current instanceof ccui.Layout)?current:ccui.Widget.\_focusedWidget;}else return ccui.Widget.prototype.findNextFocusedWidget.call(this,direction,this);}}},\_getChildWidgetByIndex:function(index){var locChildren=this.\_children;var size=locChildren.length,count=0,oldIndex=index;while(index<size){var firstChild=locChildren[index];if(firstChild&&firstChild instanceof ccui.Widget)return firstChild;count++;index++;}

var begin=0;while(begin<oldIndex){var child=locChildren[begin];if(child&&child instanceof ccui.Widget)return child;count++;begin++;}

return null;},\_isLastWidgetInContainer:function(widget,direction){var parent=widget.getParent();if(parent==null||!(parent instanceof ccui.Layout))return true;var container=parent.getChildren();var index=container.indexOf(widget);if(parent.getLayoutType()===ccui.Layout.LINEAR\_HORIZONTAL){if(direction===ccui.Widget.LEFT){if(index===0)return this.\_isLastWidgetInContainer(parent,direction);else return false;}

if(direction===ccui.Widget.RIGHT){if(index===container.length-1)return this.\_isLastWidgetInContainer(parent,direction);else return false;}

if(direction===ccui.Widget.DOWN)return this.\_isLastWidgetInContainer(parent,direction);if(direction===ccui.Widget.UP)return this.\_isLastWidgetInContainer(parent,direction);}else if(parent.getLayoutType()===ccui.Layout.LINEAR\_VERTICAL){if(direction===ccui.Widget.UP){if(index===0)return this.\_isLastWidgetInContainer(parent,direction);else return false;}

if(direction===ccui.Widget.DOWN){if(index===container.length-1)return this.\_isLastWidgetInContainer(parent,direction);else return false;}

if(direction===ccui.Widget.LEFT)return this.\_isLastWidgetInContainer(parent,direction);if(direction===ccui.Widget.RIGHT)return this.\_isLastWidgetInContainer(parent,direction);}else{cc.log("invalid layout Type");return false;}},\_isWidgetAncestorSupportLoopFocus:function(widget,direction){var parent=widget.getParent();if(parent==null||!(parent instanceof ccui.Layout))return false;if(parent.isLoopFocus()){var layoutType=parent.getLayoutType();if(layoutType===ccui.Layout.LINEAR\_HORIZONTAL){if(direction===ccui.Widget.LEFT||direction===ccui.Widget.RIGHT)return true;else return this.\_isWidgetAncestorSupportLoopFocus(parent,direction);}

if(layoutType===ccui.Layout.LINEAR\_VERTICAL){if(direction===ccui.Widget.DOWN||direction===ccui.Widget.UP)return true;else return this.\_isWidgetAncestorSupportLoopFocus(parent,direction);}else{cc.assert(0,"invalid layout type");return false;}}else return this.\_isWidgetAncestorSupportLoopFocus(parent,direction);},\_passFocusToChild:function(direction,current){if(this.\_checkFocusEnabledChild()){var previousWidget=ccui.Widget.getCurrentFocusedWidget();this.\_findProperSearchingFunctor(direction,previousWidget);var index=this.onPassFocusToChild(direction,previousWidget);var widget=this.\_getChildWidgetByIndex(index);if(widget instanceof ccui.Layout){widget.\_isFocusPassing=true;return widget.findNextFocusedWidget(direction,widget);}else{this.dispatchFocusEvent(current,widget);return widget;}}else return this;},\_checkFocusEnabledChild:function(){var locChildren=this.\_children;for(var i=0,len=locChildren.length;i<len;i++){var widget=locChildren[i];if(widget&&widget instanceof ccui.Widget&&widget.isFocusEnabled())return true;}

return false;},getDescription:function(){return"Layout";},\_createCloneInstance:function(){return new ccui.Layout();},\_copyClonedWidgetChildren:function(model){ccui.Widget.prototype.\_copyClonedWidgetChildren.call(this,model);},\_copySpecialProperties:function(layout){if(!(layout instanceof ccui.Layout))return;this.setBackGroundImageScale9Enabled(layout.\_backGroundScale9Enabled);this.setBackGroundImage(layout.\_backGroundImageFileName,layout.\_bgImageTexType);this.setBackGroundImageCapInsets(layout.\_backGroundImageCapInsets);this.setBackGroundColorType(layout.\_colorType);this.setBackGroundColor(layout.\_color);this.setBackGroundColor(layout.\_startColor,layout.\_endColor);this.setBackGroundColorOpacity(layout.\_opacity);this.setBackGroundColorVector(layout.\_alongVector);this.setLayoutType(layout.\_layoutType);this.setClippingEnabled(layout.\_clippingEnabled);this.setClippingType(layout.\_clippingType);this.\_loopFocus=layout.\_loopFocus;this.\_\_passFocusToChild=layout.\_\_passFocusToChild;this.\_isInterceptTouch=layout.\_isInterceptTouch;},forceDoLayout:function(){this.requestDoLayout();this.\_doLayout();},\_createRenderCmd:function(){if(cc.\_renderType===cc.game.RENDER\_TYPE\_WEBGL)return new ccui.Layout.WebGLRenderCmd(this);else return new ccui.Layout.CanvasRenderCmd(this);}});var \_p=ccui.Layout.prototype;\_p.clippingEnabled;cc.defineGetterSetter(\_p,"clippingEnabled",\_p.isClippingEnabled,\_p.setClippingEnabled);\_p.clippingType;cc.defineGetterSetter(\_p,"clippingType",null,\_p.setClippingType);\_p.layoutType;cc.defineGetterSetter(\_p,"layoutType",\_p.getLayoutType,\_p.setLayoutType);\_p=null;ccui.Layout.create=function(){return new ccui.Layout();};ccui.Layout.BG\_COLOR\_NONE=0;ccui.Layout.BG\_COLOR\_SOLID=1;ccui.Layout.BG\_COLOR\_GRADIENT=2;ccui.Layout.ABSOLUTE=0;ccui.Layout.LINEAR\_VERTICAL=1;ccui.Layout.LINEAR\_HORIZONTAL=2;ccui.Layout.RELATIVE=3;ccui.Layout.CLIPPING\_STENCIL=0;ccui.Layout.CLIPPING\_SCISSOR=1;ccui.Layout.BACKGROUND\_IMAGE\_ZORDER=-1;ccui.Layout.BACKGROUND\_RENDERER\_ZORDER=-2;