УДК 004

Система прогноза распространения пожаров растительности по условиям погоды

Колесников Алексей Александрович
Приамурский государственный университет имени Шолом-Алейхема
студент

Аннотация
В данной статье проведена разработка информационной системы прогнозирования распространения лесного пожара по погодным условиям.
Ключевые слова: лесной пожар, моделирование, информационная система.

Vegetation fire propagation forecast system based on weather conditions

Kolesnikov Aleksey Aleksandrovich
Sholom-Aleichem Priamursky State University
Student

Abstract
In this article, the development of an information system for predicting the spread of forest fire by weather conditions is carried out.
Keywords: forest fire, modeling, information system.

Пожарная статистика лесных пожаров в Российской Федерации в последние годы наглядно показала, что эффективный мониторинг лесных пожаров невозможен без применения современных геоинформационных технологий (ГИС). При их использовании, как правило, приоритет идёт на оперативном обнаружении тепловых аномалий по данным дистанционного зондирования и архивном картировании гарей с дополнительным использованием глобальной спутниковой навигации. Но при этом остаются открытыми вопросы прогнозирования распространения, локализации и ликвидации пожаров.

В России и за рубежом разработано несколько типов моделей распространения пожаров отдельно для каждого типа: верховые, низовые и подземные [2]. Они существенно различаются по набору исходных и расчётных данных, алгоритмам их обработки. Например, в простых методиках, основанных на эмпирических зависимостях, используются только метеорологические данные и данные о подстилающей поверхности, а результатом являются скорости распространения по 4 сторонам (фронт, фланги, тыл) с учётом времени. Более сложные модели дополнительно учитывают рельеф, состав и состояние лесного покрова, естественные и искусственные преграды и с использованием многомерных уравнений

газовой динамики определяют области сгоревших и горящих участков леса в трёхмерном представлении [1].

Прежде всего, необходимо определить язык написания. Для написания данного программного обеспечения выбор пал на язык web-программирования – "Hypertext Preprocessor" (PHP).

Главное преимущество языка PHP перед другими языками, используемых в веб-технологиях — это его простота PHP достаточно прост для изучения. PHP предоставляет веб-разработчикам возможность быстрого создания динамических веб-страниц и систем различной сложности, ориентированных на решение различных задач.

Практический характер PHP обусловлен пятью важными характеристиками:

- традиционностью;
- простотой;
- эффективностью;
- безопасностью;
- гибкостью.

Существует ещё одна «характеристика», которая делает PHP особенно привлекательным: он распространяется бесплатно! Причём с открытыми исходными кодами (Open Source). На PHP написано огромное количество сценариев различной сложности и ориентации, которые в основном, также распространяются бесплатно. Языку PHP посвящены огромное число интернет-ресурсов, на нём написаны большинство систем управления контентом (CMS), форумы, системы опросов, рассылки, и т.п.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web-программирования, по крайней мере, в ближайшее время [3].

После определения с языком программирования необходимо выбрать среду разработки системы. Выбор среды для данной системы пал на Framework Yii2.

Yii2 – это объектно-ориентированный компонентный фреймворк, реализующий парадигму MVC (Model-View-Controller).

Преимущества данного фреймворка над подобными отмечены и профессиональным сообществом разработчиков, и аналитиками. От аналогов Yii2 отличает:

1. Относительно простая базовая архитектура для организации кода. Это подразумевает высокую степень универсальности использования фреймворка и простоту работы с ним.
2. Производительность. Yii2 является одной из самых быстрых структурных оболочек.
3. Способность взаимодействия практически со всеми типами баз данных.
4. Возможность подключения сторонних классов, библиотек и расширений.

5. Применение различных вариантов кэширования.

6. Генерация PHP-кода.

7. Существование большого объёма русскоязычной документации и активного интернет-сообщества пользователей и разработчиков.

Главная особенность использования Yii2 заключается в строгом соответствии заданным стандартам и логичности архитектуры. Это подразумевает возможность технического сопровождения ресурса силами штатного администратора-универсала, но только, если сайт создавался командой профессионалов. [4]

В среде web-программирования Yii2 разработка и модификация компонентов модели (Model), представления (View) и контролёра (Controller) может осуществляться независимо друг от друга.

Листинг файла набора подключаемых файлов, скриптов и стилей (assets\AppAsset.php):

```php
<?php
namespace app\assets;
use yii\web\AssetBundle;
class AppAsset extends AssetBundle
{
    public $basePath = '@webroot';
    public $baseUrl = '@web';
    public $css = [
        'css/site.css',
    ];
    public $js = [
        'js/map.js'
    ];
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap\BootstrapAsset',
    ];
```

Рисунок 1 - Листинг файла AppAsset.php

Листинг файла контролёра, отвечающим за обработку запросов, а также генерацию ответов (controllers\SiteController.php):

```php
<?php
namespace app\controllers;
use app\models\Grid;
use app\models\InputDataForm;
use Yii;
use yii\web\Controller;
class SiteController extends Controller
{
    /**
     *  главная страница
     *
     * @return string
     */
    public function actionIndex()
    {
        $grid = new Grid();
        $model = new InputDataForm();
        if ($model->load(Yii::$app->request->post()) && $model->validate()) {
            $model->calculateEndPoint();
            $grid->line = $model->line;
        }
        Yii::$app->session->set('model', $model->toArray());
        return $this->render('index', ['model' => $model, 'grid' => $grid]);
    }
    /**
     *
     * action для загрузки сетки динамически
     * @param $time
     * @return string
     */
    public function actionLoadAjax($time)
    {
        $grid = new Grid();
        /** @var InputDataForm $model */
        if ($model = new InputDataForm()) {
            if ($model->load(Yii::$app->request->get()))
                $model->time = $time;

            $model->calculateEndPoint();
            $grid->line = $model->line;
            // выводим сетку на экран
            return $this->renderAjax('grid', ['grid' => $grid]);
        }
    }
```

Рисунок 2 - Листинг файла SiteController.php

В структуре Framework Yii2 файлы, отвечающие за представление данных конечным пользователям, располагаются в папке views. Одним из таких файлов-настроек называется index.php. Он является главной страницей каждого разрабатываемого web-ресурса. Листинг настроек для нашей системы указаны на рисунках 3 и 4.

```
<?php
$this->title = 'My Yii Application';
use app\models\InputDataForm;
use yii\helpers\Html;
use yii\widgets\ActiveForm; ?>
    <style>
        #map {
            width: 500px;
            height: 500px;
            padding: 0;
            margin: 0;
        }
        a {
            color: #04b; /* Цвет ссылки */
            text-decoration: none; /* Убираем подчеркивание у ссылок */
        }
        a:visited {
            color: #04b; /* Цвет посещённой ссылки */
        }
        a:hover {
            color: #f50000; /* Цвет ссылки при наведении на нее курсора мыши */
        }
        table {
            width: 100%;
            height: 100%;
        }
        td {
            border: grey 1px solid;
        }
        td.fire {
            background: orange;
            opacity: 0.8;
        }
        td.red {
            background: orange;
            opacity: 0.8;
        }
        td:hover {
            background: red;
            opacity: 0.8;
        }
    </style>
```

Рисунок 3 - Листинг файла index.php

```
    <div class="container">
        <div class="row">
            <div class="col-lg-6">
                <div style="width: 500px; height: 500px;position: relative;">
                    <div id="map" style="width: 100%; height: 100%; position: absolute;"></div>
                    <div id="grid" style="position: absolute; width: 100%; height: 100%;">
                        <?php echo $grid->render(true); ?>
                    </div>
                </div>
            </div>
            <div class="col-lg-5">
                <div>
                    <?php $form = ActiveForm::begin(['id' => 'form']); ?>
                    <?= $form->field($model, 'xStart')->textInput() ?>
                    <?= $form->field($model, 'yStart')->textInput() ?>
                    <?= $form->field($model, 'rain_force')->dropDownList([0 => "нет"] + InputDataForm::mapRainForce()) ?>
                    <?= $form->field($model, 'wind_force') ?>
                    <?= $form->field($model, 'wind_direction') ?>
                    <?= $form->field($model, 'temperature')->dropDownList(InputDataForm::mapTemperature()) ?>
                    <?= Html::submitButton('Сохранить данные', ['class' => 'btn btn-primary', 'name' => 'contact-button']) ?>
                    <?php ActiveForm::end(); ?>
                    <?= Html::button('Старт', ['class' => 'btn btn-success btn-lg', 'id' => 'start']) ?>
                    <?= Html::button('Пауза', ['class' => 'btn btn-success btn-lg', 'id' => 'pause']) ?>
                    <?= Html::button('Сбросить', ['class' => 'btn btn-danger btn-lg', 'id' => 'reset']) ?>
                    <?= Html::button('Передвинуть карту', ['class' => 'btn btn-primary btn-lg', 'id' => 'toggle-grid']) ?>
                </div>
            </div>
        </div>
    </div>
<?php
$js = <<<JS
JS;
```

Рисунок 4 - Окончание листинга файла index.php

Помимо папки видов (views), для хранения файлов, относящихся к внешнему виду страниц, также используется папка web, в которой находятся все файлы стилей, картинок, скриптов и т.д. Для данной системы в папке web\js был создан файл map.js, отвечающий за рисование на карте распространения пожара. Листинг файла указан на рисунке 5.

```
try {
    ymaps.ready(function () {
        var myMap = new ymaps.Map('map', {
            center: [55.751574, 37.573856],
            zoom: 9,
            behaviors: ['default', 'scrollZoom']
        }, {
            searchControlProvider: 'yandex#search'
        })
    });
} catch (e) {
    console.log('map cannot be load');
}
$(document).on('click', '#grid td', function () {
    $('#xstart').val($(this).data('row'));
    $('#ystart').val($(this).data('col'));
    $(this).addClass('red');
});
var timer = null;
var timer_start = false;
$('#toggle-grid').click(function () {
    $('#grid').toggleClass('hidden');
});
$('#start').on('click', function () {
    timer_start = true;
    var time = 0;
    console.log($('#ystart').val());
    console.log($('#xstart').val());
    timer = setInterval(function () {
       time++;
        $('#start').text(time + ' часов');
        $('#grid').load('/site/load-ajax?time=1&' + $('#form').serialize());
    }, 3000);
});
$('#pause').on('click', function () {
    if (timer_start === true) {
        timer_start = false;
        clearInterval(timer);
    }
});
$('#reset').on('click', function () {
    if (timer_start === true) {
        $('#start').text('START');
        time = 0;
        timer_start = false;
        clearInterval(timer);
    }
    $('#grid').load('/site/clear');
});
```

Рисунок 5 - Листинг файла map.js

Также в структуре фреймворка есть папка models, которая содержит классы моделей. В данной папке были созданы файлы Grid.php, InputDataForm.php, LineSegment.php, Point.php и Square.php. Эти файлы будут отвечать за функции системы, например, файл InputDataForm.php будет отвечать за клеточное закрашивание карты, исходя из введённых на главной

странице параметров погоды. Листинг данных файлов указан на рисунках 6-13.

```php
<?php
namespace app\models;
use app\models\geo\Point;
use app\models\geo\Square;
use yii\helpers\Html;
class Grid
{
    // сохраняем пересеченную сетку в сессию
    public function storeToSession($cell)
    {
        $gridCells = $this->getFromSession();
        array_push($gridCells, $cell);
        \Yii::$app->session->set('gridCells', array_unique($gridCells));
    }
    // очищаем данные сессии
    public function clearStorage()
    {
        \Yii::$app->session->set('gridCells', []);
        \Yii::$app->session->set('xStart', null);
        \Yii::$app->session->set('yStart', null);
        \Yii::$app->session->set('lastTime', null);
    }
    // берем данные из сессии
    public function getFromSession()
    {
        return \Yii::$app->session->get('gridCells', []);
    }
     // проверка, что данная клетка уже была поражена огнем
    public function isInSession($cell)
    {
        return in_array($cell, \Yii::$app->session->get('gridCells', []));
    }
    const SCALE = 10; // px in cell
    public $from = 1;
    public $to = 50;
    public $firedCells = [];
    public $line;
    //  метод возвращает сетку с отмеченными оранжевым цветом клетками
    public function render($skip = false)
    {
```

Рисунок 6 - Листинг файла Grid.php

```
        $trs = [];
        foreach (range($this->from * self::SCALE, $this->to * self::SCALE, self::SCALE) as $row) {
            $tds = [];
            foreach (range($this->from * self::SCALE, $this->to * self::SCALE, self::SCALE) as $cell) {
                $class = '';
                //  заносим точки
                $point1 = new Point($row - 1 * self::SCALE, $cell - 1 * self::SCALE);
                $point2 = new Point($row - 1 * self::SCALE, $cell);
                $point3 = new Point($row, $cell);
                $point4 = new Point($row, $cell - 1 * self::SCALE);
                 // создаем квадрат из точек
                $square = new Square($point1, $point2, $point3, $point4);
                // проверяем на пересечение или ранее сохраненную ячейку в сессию
                if (($this->line && $square->isIntersectLine($this->line) && !$skip) || ($this->isInSession("$row:$cell"))) {
                    $this->storeToSession("$row:$cell");
                    $class = 'red';
                }
                $tds[] = Html::tag('td', '', ['class' => $class, 'data' => ['row' => $row, 'col' => $cell]]);
            }
            $trs[] = Html::tag('tr', implode('', $tds));
        }
         // выводим html таблицу
        return Html::tag('table', implode("", $trs));
    }
}
```

Рисунок 7 - Окончание листинга файла Grid.php

```
<?php
namespace app\models;
use app\models\geo\LineSegment;
use yii\base\Model;
/**
 * InputDataFrom is the model to load data to calculate next point of fire
 * @property float|int speed
 */
class InputDataForm extends Model
{
    const PI = 3.14;
    const METERS_IN_PIXELS = 1000;
    public $wind_force = 0;
    public $wind_direction = 0;
    public $rain_force = 0;
    public $speed;
    public $xStart = 0;
    public $yStart = 0;
    public $temperature = 0;
    public $xEnd;
    public $yEnd;
    public $line;
    public $time = 1; // hour
    public $timeStart = 0;
    public function setNextPoint()
    {
        \Yii::$app->session->set('xStart', $this->xEnd);
        \Yii::$app->session->set('yStart', $this->yEnd);
        \Yii::error('SET LAST TIME ='.\Yii::$app->session->get('lastTime'));
    }
    public function getNextPoint()
    {
        $this->xStart = \Yii::$app->session->get('xStart') ?: ($this->xStart - Grid::SCALE / 2);
        $this->yStart = \Yii::$app->session->get('yStart') ?: ($this->yStart - Grid::SCALE / 2);
        \Yii::error('GET LAST TIME ='.\Yii::$app->session->get('lastTime'));
    }
    public static function mapRainForce()
    {
        return [
            1 => "Слабый",
            2 => "Средний",
            3 => "Сильный"
        ];
    }
```

Рисунок 8 – Листинг файла InputDataForm.php

```php
public function formName()
{
    return '';
}
public static function mapTemperature()
{
    return [
        0 => "0-10 градусов",
        1 => "11-20 градусов",
        2 => "21+ градусов"
    ];
}
/**
 * @return array the validation rules.
 */
public function rules()
{
    return [
        [['yStart', 'xStart', 'yEnd', 'xEnd', 'rain_force'], 'number'],
        [['wind_direction'], 'number', 'min' => 0, 'max' => 360],
        [['wind_force'], 'number', 'min' => 0, 'max' => 50],
        [['temperature'], 'number', 'min' => -40, 'max' => 50],
        [['time'], 'integer', 'min' => 0, 'max' => 200],
    ];
}
/**
 * @return array customized attribute labels
 */
public function attributeLabels()
{
    return [
        'yStart' => 'Координата пожара,Y',
        'xStart' => 'Координата пожара,X',
        'rain_force' => 'Интенсивность дождя',
        'wind_force' => 'Скорость ветра, м/с',
        'wind_direction' => 'Направление ветра, градусы(0-360)',
        'time' => 'Продолжительность пожара',
        'temperature' => 'Температура воздуха, градусы Цельсия',
    ];
}
```

Рисунок 9 - Продолжение листинга файла InputDataForm.php

```php
    public function calculateEndPoint()
    {
        // инициализация точки окончания если мы решили продолжить
        $this->getNextPoint();
        // скорость распространения огня по умолчанию
        $speed = 0.2;
        // скорость распространения огня в зависимости от скорости ветра
        if ($this->wind_force) $speed = $speed + floor($this->wind_force / 3);
        // делаем поправку на дождь в зависимости от его силы
        if ($this->rain_force) {
            $speed = $speed - $this->rain_force;
        }
        // если сильный дождь, то огонь тушится сам
        if ($this->rain_force > 2) {
            $speed = 0;
        }
        // если скорость ветра получилась отрицательная, то ставим 0
        if ($speed < 0) {
            $speed = 0;
        }
        // делаем поправку на температуру воздуха
        if ($this->temperature) {
            $speed = $speed + $this->temperature;
        }
        $this->speed = $speed;
        // вычитаем время прошлого отрезка, чтобы вычислить расстояние которое пройдет огонь с прошлого часа
        $this->time = $this->time - $this->timeStart;
        // вычисляем координаты, куда придет огонь через определеное время с вычисленной скоростью
        $this->xEnd = $this->xStart + $this->time * $speed * cos(self::PI * (180 - floatval($this->wind_direction)) / 180);
        $this->yEnd = $this->yStart + $this->time * $speed * sin(self::PI * (180 - floatval(180 - $this->wind_direction)) / 180);
        // сохраняем в сессии точку, где закончился гореть огонь
        $this->setNextPoint();
        $this->line = new LineSegment($this->xStart, $this->xEnd, $this->yStart, $this->yEnd);
    }
}
```

Рисунок 10 - Окончание листинга файла InputDataForm.php

```php
<?php
namespace app\models\geo;
class LineSegment
{
    public $x1;
    public $y1;
    public $x2;
    public $y2;
    public function __construct($x1, $x2, $y1, $y2)
    {
        $this->x1 = $x1;
        $this->y1 = $y1;
        $this->x2 = $x2;
        $this->y2 = $y2;
    }
    public function render()
    {
        return round($this->x1)."-".round($this->y1)."<br>".round($this->x2)."-".round($this->y2);
    }
    /**
     * @return bool
     * @param self $line
     */
    public function isIntersectLine(LineSegment $line)
    {
        $common = ($this->x2 - $this->x1) * ($line->y2 - $line->y1) - ($this->y2 - $this->y1) * ($line->x2 - $line->x1);
        if ($common == 0) {
            return false;
        }
        $rH = ($this->y1 - $line->y1) * ($line->x2 - $line->x1) - ($this->x1 - $line->x1) * ($line->y2 - $line->y1);
        $sH = ($this->y1 - $line->y1) * ($this->x2 - $this->x1) - ($this->x1 - $line->x1) * ($this->y2 - $this->y1);
        $r = $rH / $common;
        $s = $sH / $common;
        if ($r >= 0 && $r <= 1 && $s >= 0 && $s <= 1) {
            return true;
        } else {
            return false;
        }
    }
}
```

Рисунок 11 - Листинг файла LineSegment.php

```php
<?php
namespace app\models\geo;
class Point
{
    public $x;
    public $y;
    /**
     * Point constructor.
     * @param $x
     * @param $y
     */
    public function __construct($x, $y)
    {
        $this->x = $x;
        $this->y = $y;
    }
}
```

Рисунок 12 – Листинг файла Point.php

```php
<?php
namespace app\models\geo;
class Square
{
    public $point1;
    public $point2;
    public $point3;
    public $point4;
    public $upLine;
    public $downLine;
    public $leftLine;
    public $rightLine;
    /**
     * функция конструктор квадрата из точек
     * @param $point1 Point
     * @param $point2 Point
     * @param $point3 Point
     * @param $point4 Point
     */
    public function __construct($point1, $point2, $point3, $point4)
    {
        $this->point1 = $point1;
        $this->point2 = $point2;
        $this->point3 = $point3;
        $this->point4 = $point4;
        // создаем грани из координат
        $this->composeLines();
    }
    public function render()
    {
        return printf('%s-%s-%s-%s  %s-%s-%s-%s', $this->point1->x, $this->point1->y, $this->point2->x, $this->point2->y, $this->point3->x, $this->point3->y, $this->point4->x, $this->point4->y);
    }
    /**
     *  метод создает грани квадрата из его координат
     */
    private function composeLines()
    {
        $this->upLine = new LineSegment($this->point1->x, $this->point2->x, $this->point1->y, $this->point2->y);
        $this->leftLine = new LineSegment($this->point2->x, $this->point3->x, $this->point2->y, $this->point3->y);
        $this->downLine = new LineSegment($this->point3->x, $this->point4->x, $this->point3->y, $this->point4->y);
        $this->rightLine = new LineSegment($this->point4->x, $this->point1->x, $this->point4->y, $this->point1->y);
    }
    /**
     * функция возвращает ИСТИНА, если любая из граней пересекает основную линию огня
     * @param $line
     * @return bool
     */
    public function isIntersectLine($line)
    {
        return $this->upLine->isIntersectLine($line)
            || $this->leftLine->isIntersectLine($line)
            || $this->downLine->isIntersectLine($line)
            || $this->rightLine->isIntersectLine($line);
    }
}
```

Рисунок 13 – Листинг файла Square.php

Используя данный для web-программирования язык был разработан прототип программного продукта. На рисунке 14 изображён внешний вид разработанной системы.

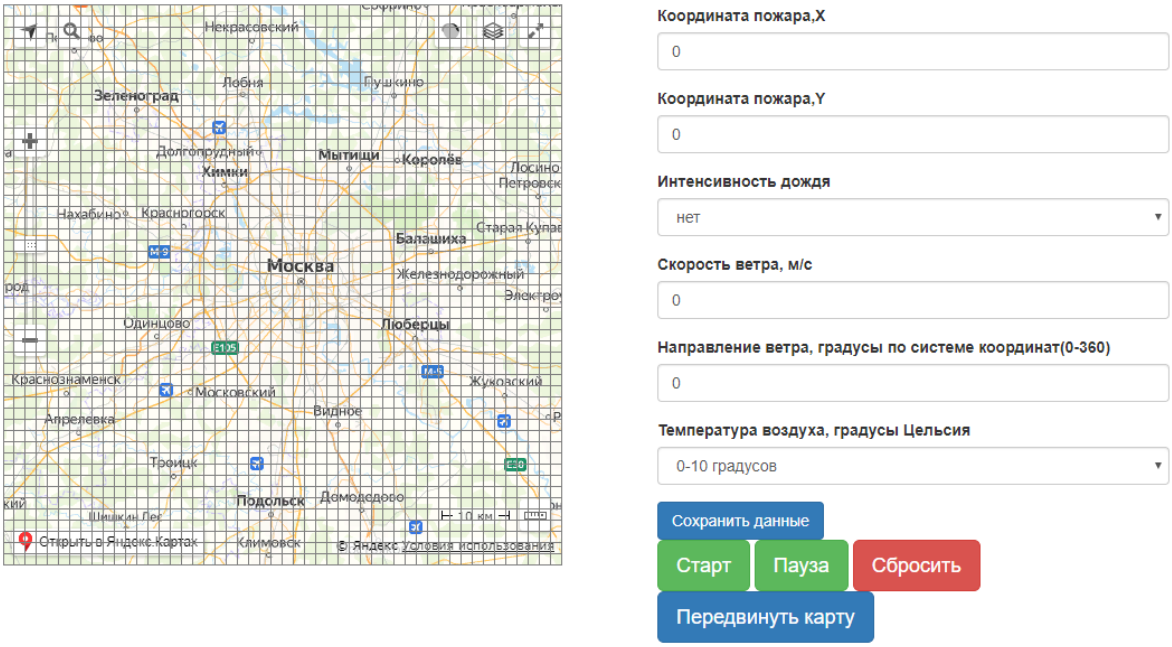

Рисунок 14 – Внешний вид разработанной системы

Для начала работы системы требуется указать место, в котором начнётся симуляция пожара. Для этого необходимо нажать на кнопку «Передвинуть карту» и пролистать на поле слева карту до нужной точки. При этом также имеется возможность увеличить либо же уменьшить масштаб карты. Пример передвинутой карты показан на рисунке 15.

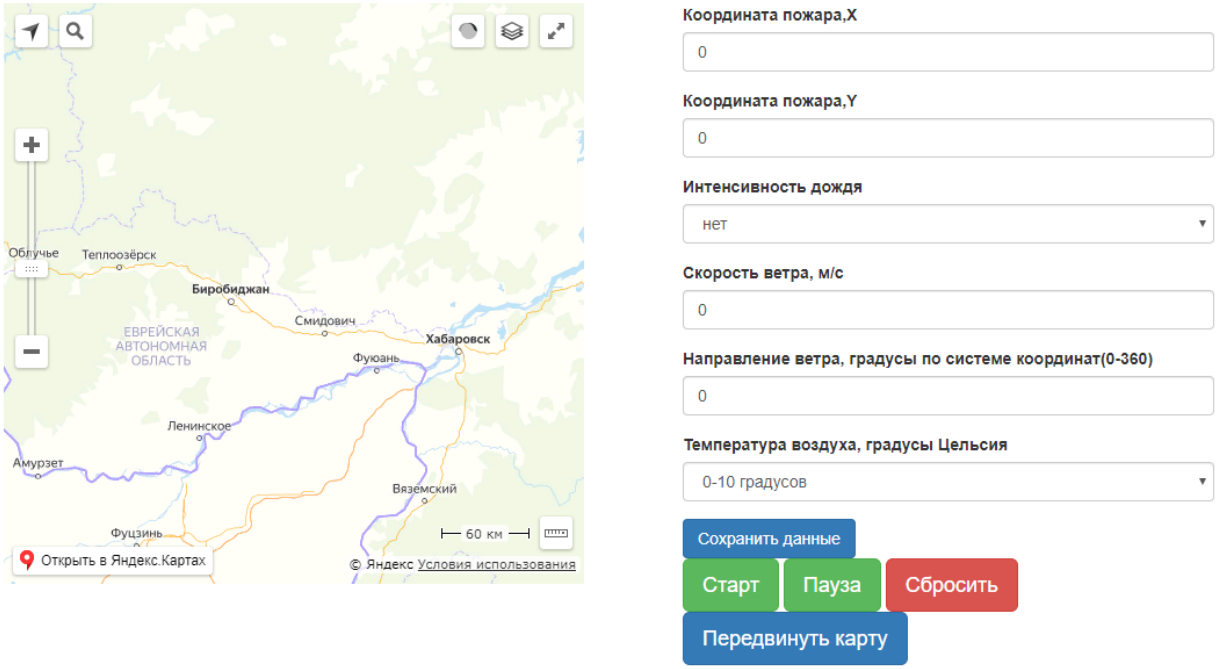

Рисунок 15 - Пример сдвинутой карты

После перемещения карты в желаемую область нужно снова нажать на кнопку «Передвинуть карту», чтобы сменить режим карты на симуляцию пожара. Теперь необходимо указать начальную точку распространения пожара, указав её либо на сетке карты, либо введя числовые значения в

координаты X и Y. Дальше нужно указать желаемые для начала симуляции погодные условия: интенсивность дождя (при наличии), скорость ветра, направление ветра и температуру воздуха. Пример установленных настроек показан на рисунке 16.

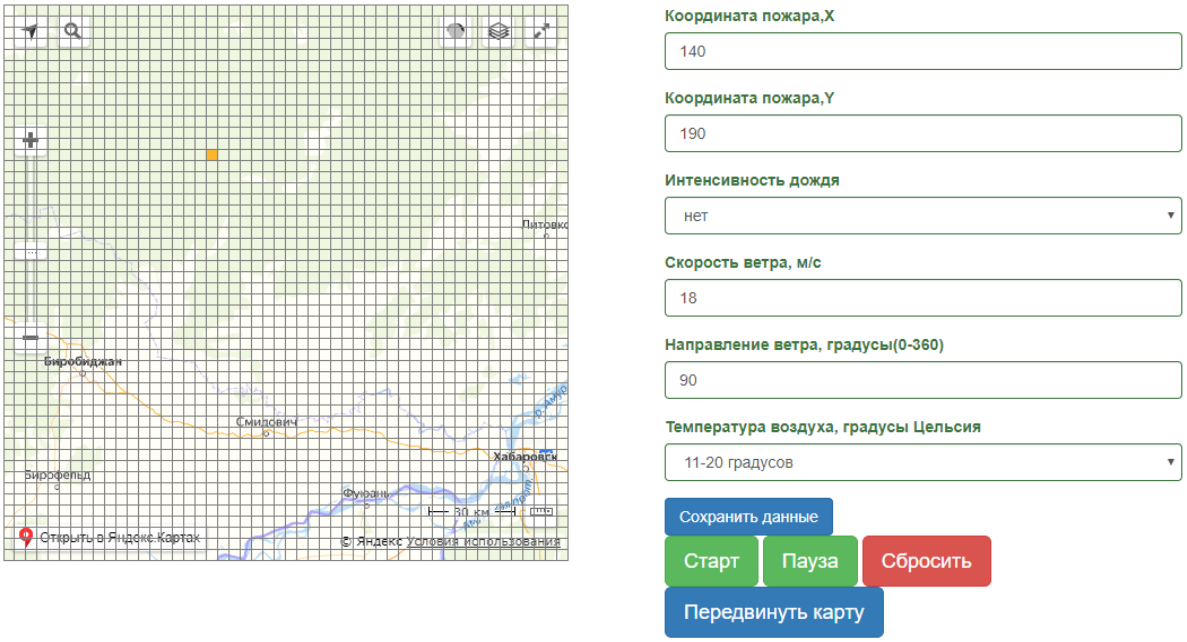

Рисунок 16 - Настройка параметров симуляции

После установки начальных параметров нужно нажать кнопку «Старт» для инициации начала симуляции. Спустя определённое количество времени в зависимости от введённых значений будет закрашиваться определённое количество клеток сетки на карте. Пример распространения пожара (закрашивания клеток) показан на рисунке 17.

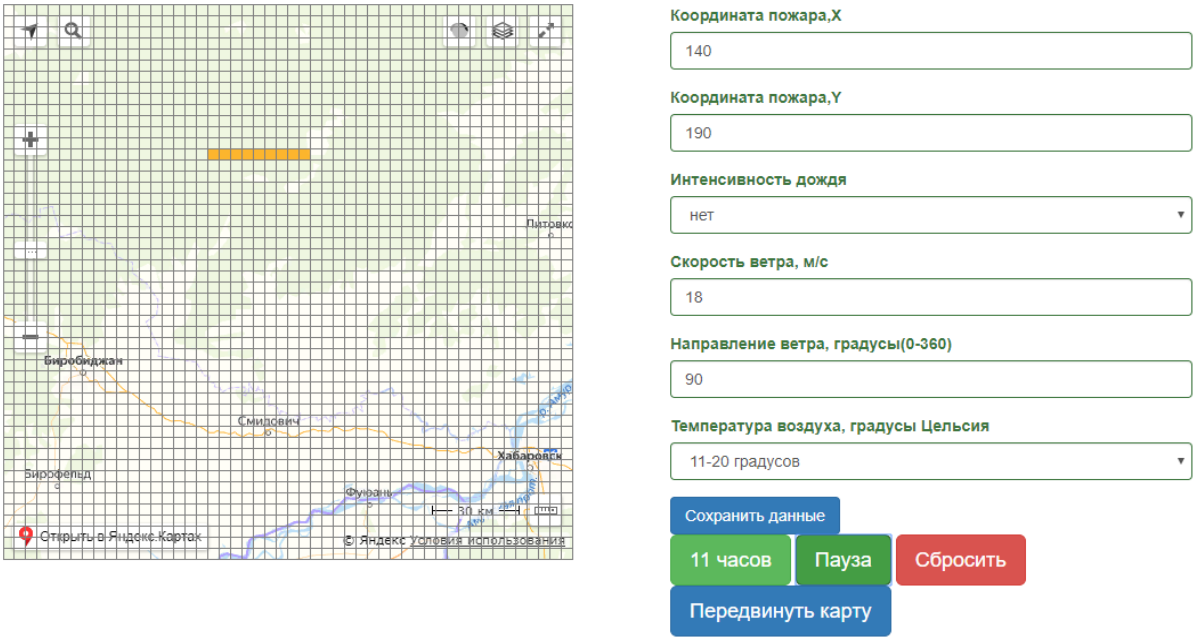

Рисунок 17 – Пример распространения пожара

Далее, при желании можно по кнопке «Пауза» приостановить симуляцию и либо сбросить результат прохождения симуляции, нажав на кнопку «Сбросить», либо установить новые значения параметров симуляции. Если всё же были введены новые значения, то можно продолжить симуляцию нажав кнопку «Старт», при этом счётчик пройденного пути начнётся сначала. Пример продолжения симуляции с новыми значениями показан рисунке 18.

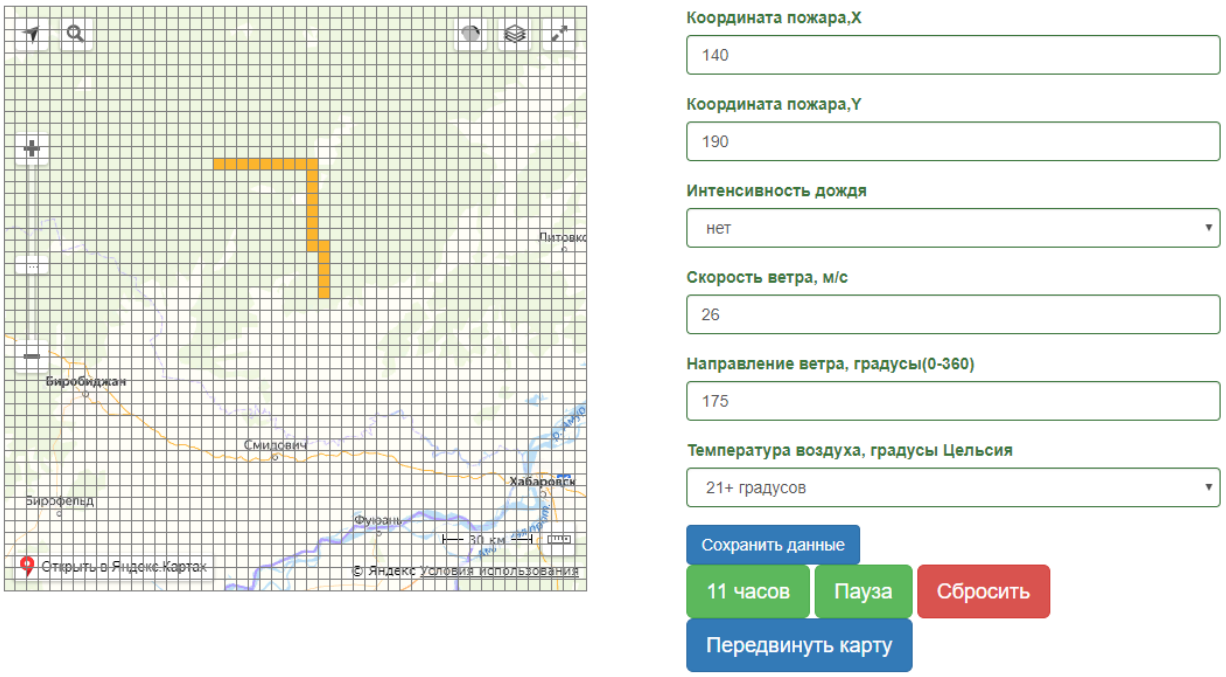

Рисунок 18 – Пример продолжения симуляции

Нажав на кнопку «Сохранить данные», система сохраняет введённые последние параметры симуляции и пройденный по сетке карты путь распространения пожара, сбрасывая время распространения пожара и саму карту на базовые значения. Результаты срабатывания кнопки показаны на рисунке 19.

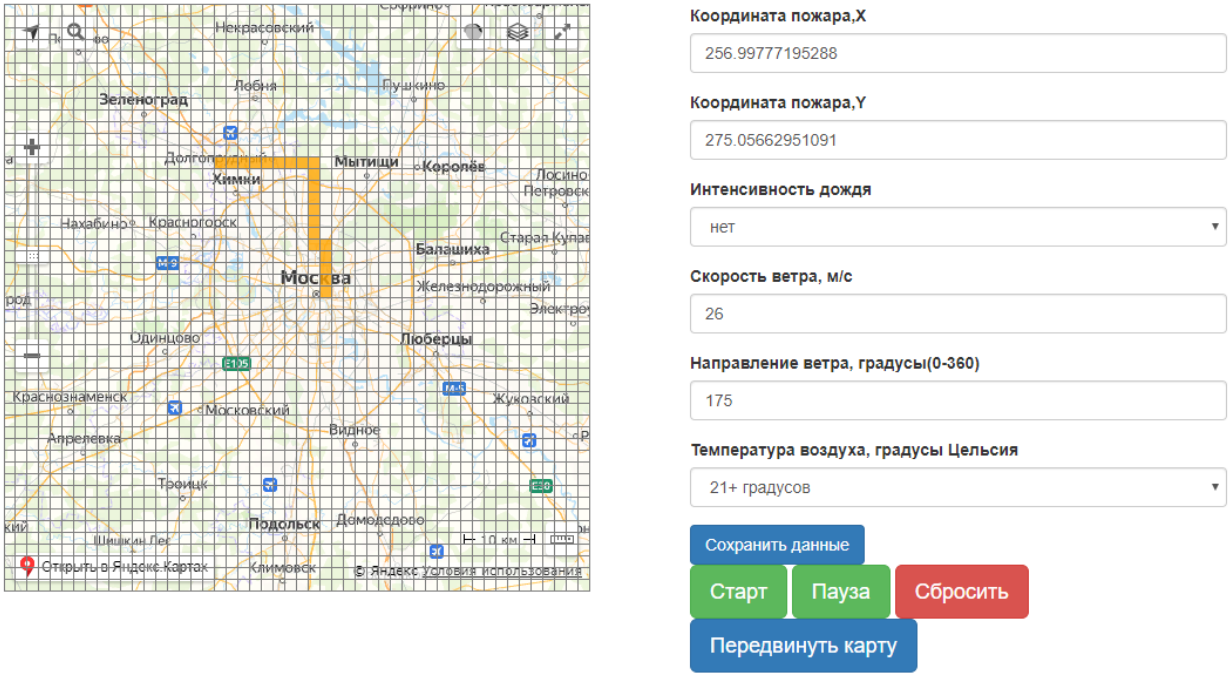

Рисунок 19 - Результат нажатия кнопки «Сохранить данные»

Результатом статьи является разработанная система моделирования прогнозирования распространения лесного пожара по погодным условиям.

Библиографический список

1. Гиниятов А.А., Кугуракова В.В., Якушев Р.С. Разработка симуляционного приложения для моделирования лесных пожаров с учётом погодных условий и формы ландшафта // Электронные библиотеки. 2016. №3. С. 180-192.
2. Гришин А. М. Математическое моделирование лесных пожаров и новые способы борьбы с ними. Новосибирск: Наука 1992, с 407
3. FAQ. Общие вопросы по PHP // PHP.SU URL: http://www.php.su/faq/?basic (дата обращения: 22.04.2020).
4. Преимущества разработки сайта на фреймворке Yii2 // Яндекс Дзен URL: https://zen.yandex.ru/media/id/5bc790f4c0bf6300aaa944ad/preimuscestva-razrabotki-saita-na-freimvorke-yii2-5bef584ae9397500ab3a07c4?utm_source=serp