Министерство образования Хабаровского края
Краевое государственное бюджетное образовательное учреждение
среднего профессионального образования
«Хабаровский колледж межотраслевых технологий и сферы обслуживания»

Методическое пособие
Программирование игр в среде 1С:Предприятие

ОП.04 Основы алгоритмизации и программирования
ПМ05. Проектирование и разработка информационных систем
для специальности
09.02.07 «Информационные системы и программирование»

Разработано преподавателем
комиссии информационных
дисциплин
Мазур Т.В.

Хабаровск, 2025

СОДЕРЖАНИЕ

Пояснительная записка

Обычно 1С:Предприятие воспринимается как платформа для разработки бизнес-приложений: учёт закупок и продаж, работа с контрагентами и номенклатурой, кассовые операции — именно этим мы, как правило, обучаем студентов, формируя у них навыки автоматизации хозяйственной деятельности предприятий и бизнес-процессов.

Однако в учебном процессе полезно делать и «творческие паузы» — отход от рутины стандартных задач помогает лучше усвоить материал и пробудить интерес к разработке. Одним из таких нестандартных, но очень эффективных приёмов является программирование игр в среде 1С. Это не только развивает алгоритмическое мышление, но и позволяет взглянуть на платформу с новой стороны — как на гибкий инструмент для решения любых задач, в том числе и игровых.

Написание игр в среде 1С:Предприятие — это не просто развлечение. Это обучающий инструмент, который помогает студентам освоить ключевые навыки разработки в 1С через практическую, наглядную и мотивирующую задачу.

Вот основные причины, почему студентам полезно писать игры на 1С.

1. **Повышение мотивации и вовлечённости**

Игры — это зримый результат.

Считаю, что разработка игр делает обучение интересным, формируя при этом навыки разработки в 1С. Студент видит, как его код «оживает»: клики, ходы, победы — всё работает здесь и сейчас.

Это особенно важно на начальном этапе, когда абстрактные «документы» и «регистры» кажутся скучными.

2. Освоение синтаксиса и логики языка 1С

Различные игры могут требовать:

- работы с переменными, условиями, циклами;
- использования массивов, соответствий, таблиц значений;

- понимания типов данных и операций со строками и т.д.

Через игру студент «чувствует» язык, а не просто заучивает правила.

3. Работа с формами и элементами управления

Разрабатывая игры, студент может научиться:

- создавать и настраивать формы,
- добавлять кнопки, табличные документы, переключатели,
- обрабатывать события (нажатие, выбор ячейки),
- работа с графическими элементами;
- управлять видимостью и доступностью элементов.

Это базовые навыки для любой конфигурации — от отчётов до сложных бизнес-процессов.

4. Подготовка к реальным задачам

Хотя 1С — платформа для бизнес-приложений, многие задачи похожи на игровые:

- проверка условий («можно ли провести документ?»);
- блокировка/разблокировка полей;
- визуальная обратная связь (подсветка ошибок);
- ведение счёта (статистика, аналитика).

Игра — это микромодель бизнес-логики в упрощённой, понятной форме.

5. Развитие алгоритмического мышления

В некоторые игры требуется добавить логику игрока-бота, который будет стремиться победить человека. От студента будет требоваться написать несложный вариант ИИ. Например, игра «Крестики-нолики» требует:

- анализа состояния поля;
- принятия решений по приоритетам (выиграть → защититься → занять центр);
- перебора вариантов.

Это введение в логику принятия решений — основа автоматизации бизнес-процессов.

Было замечено, что олимпиада по программированию от компании 1С делает особый акцент на алгоритмическом мышлении. Задания регионального тура направлены не столько на знание специфики платформы или объектов конфигурации, сколько на умение чётко формулировать задачу, выстраивать логику её решения и эффективно реализовывать алгоритм.

Именно поэтому в подготовке важно развивать не только технические навыки работы в 1С, но и способность думать, анализировать и находить оптимальные пути — ведь именно это лежит в основе настоящего программирования.

По моему опыту разработка игр в этом сильно помогает. Но специфику 1С тоже можно учитывать. Например, результаты игроков можно записывать в справочник (журнал, регистр). Проводить статистику, анализируя табличные данные, что отражает специфику 1С и т.д.

Данное методическое пособие используется в рамках учебной практики по модулю ПМ.05 Разработка информационных систем на специальности 09.02.07 «Информационные системы и среды». Учебная практика завершает 72-часовой курс «Разработка в среде 1С». Также вижу возможность применить материалы пособия в кружковой работе, организации проектной деятельности учащихся, на дисциплине «Основы алгоритмизации и программирования».

Каждая практическая работа пособия содержит вопросы, которые надо обсудить перед разработкой: правила игры, особенности компьютерного варианта игры, организацию действий игрока и бота (если его участие предусмотрено), варианты победы или поражения, хранения информации, переменные, элементы интерфейса, алгоритмы. исключения и т.п. Представлены этапы разработки интерфейса и программирования. Особое внимание уделено объяснению назначения и алгоритмам программ.

Методическое пособие можно использовать в различных форматах учебного процесса.

Во-первых, оно подойдёт для самостоятельной работы — как студентам, стремящимся углубить свои навыки программирования в среде

1С:Предприятие, так и начинающим преподавателям, осваивающим методику обучения разработке на платформе 1С.

Во-вторых, пособие может быть эффективно применено в рамках урочной деятельности. В нём подробно описан ход первого занятия по созданию игры «Крестики-нолики»: от вводной беседы, формирующей общее понимание задачи, до пошаговых инструкций по реализации. Особое внимание уделено этапу самостоятельного развития проекта: студентам предлагаются задания на расширение функциональности игры, а также вопросы для рефлексии, направленные на осмысление общих принципов проектирования игровых приложений и размышление о том, какие ещё игры могут быть реализованы в 1С.

В-третьих, преподаватель может использовать материал как основу для индивидуальных или групповых проектов. То есть не выдавать указания с готовым годом, а использовать в качестве вспомогательного материала.

При возникновении трудностей учащимся могут быть предложены фрагменты готовой разработки или общие ориентиры, но ключевым остаётся принцип активного конструирования знаний: никто не научится программировать, только наблюдая за чужим кодом. Поэтому студентам рекомендуется сначала выполнить одну–две практические работы по подробным указаниям, а затем перейти к созданию собственного проекта.

В рамках учебной практики учащимся предлагаются темы игр, сгруппированные по уровню сложности, однако им предоставляется возможность предложить и согласовать собственный вариант.

Очень важным считаю, когда студенты делятся своими знаниями и достижениями со своими одногруппниками. Лучшие работы после защиты оформляются как методические материалы для будущих поколений обучающихся. Так, в данное пособие включена разработка студента группы ИС-21 Лосева Артёма — карточная игра «21», ставшая ярким примером творческого подхода к освоению платформы 1С.

При разработке методического пособия в качестве эксперимента я попыталась создать игру «Виселица» с помощью нейросетей. надо признать, что было трудно реализовать эту задачу, так как по задумке нужно было использовать изображения. Нейросеть предлагала варианты, которые содержали ошибки, неправильные конструкции (например, предлагала использовать алгоритмическую структуру Выбор, которая в 1С отсутствует), упоминала свойства, которые отсутствуют в конфигурации и т.д. В тоже время, всё что касается работы с простыми объектами конфигурации, со строками, стандартными функциями нейросеть делает легко.

Убедившись, что разработка данной игры даже с использованием нейросети не такое лёгкое дело, что может дать необходимый опыт учащимся. я предложила на одну из домашних работ – воспользоваться для программирования искусственным интеллектом, потренироваться с составлением правильных запросов и посмотреть, как он справляется с поставленными задачами.

Программирование игры «Крестики-нолики»

Цель: разработать игру «Крестики-нолики» с возможностью играть «с компьютером». Совершенствование навыков программирования в 1С.

План занятия

1. Постановка целей (фронтальная беседа).
2. Актуализация знаний. Обсуждение разработки (фронтальная беседа).
3. Выполнение практического задания.
4. Самостоятельная работа.
5. Подведение итогов, рефлексия.

Ход занятия

Сегодня мы отступим от привычного сценария, где 1С — это склады, накладные, контрагенты и регламентированные отчёты. Да, безусловно, **1С:Предприятие** — это мощная платформа для автоматизации бизнеса. Но она — **гораздо больше**.

Платформа 1С обладает гибкой архитектурой, развитым языком программирования и богатыми возможностями работы с интерфейсом. А это значит, что на ней можно решать **не только учётные, но и творческие, логические, даже игровые задачи.**

Сегодня мы начнём с классики — с игры **«Крестики-нолики»**. На первый взгляд, это просто развлечение. Но за этой простотой скрывается **интересная задача для программиста!**

Для постановки задач необходимо ответить на следующие вопросы (фронтальная беседа со студентами).

1. Постановка задачи

- Какие правила игры в «Крестики-нолики»? Игроки по очереди ставят на свободные клетки поля 3×3 знаки (один всегда крестики, другой всегда нолики). Первый, выстроивший в ряд 3 своих фигуры по вертикали,

горизонтали или большой диагонали, выигрывает. Если игроки заполнили все 9 ячеек и оказалось, что ни в одной вертикали, горизонтали или большой диагонали нет трёх одинаковых знаков, партия считается закончившейся вничью. Первый ход делает игрок, ставящий крестики.

- Сколько игроков участвует в классической версии? А в нашей сколько будет?
- Какие возможны исходы игры? (победа, поражение, ничья)
- Как компьютер может «думать»? Что значит «искусственный интеллект» в такой простой игре?

2. На этапе проектирования структуры необходимо обсудить

- Как представить игровое поле в 1С? Какой элемент управления подойдёт? Как с ним работать?
- Нужно ли хранить состояние поля? Почему да или нет?
- Как обозначить пустую ячейку, крестик и нолик?
- Как определить, что ячейка уже занята?

3. На этапе реализации логики хода

- Кто делает первый ход: игрок или компьютер? Как это настроить?
- Что должно произойти сразу после клика игрока по ячейке?
- Как запретить делать ходы после окончания игры?
- Как передать ход компьютеру после хода игрока?

4. На этапе разработки логики ИИ (компьютера)

- Какой самый «умный» ход может сделать компьютер?
- В каком порядке компьютер должен принимать решения? (Подсказка: выиграть → не проиграть → занять выгодную позицию)
- Почему центральная клетка (Б2) считается самой сильной?
- Что делать, если все клетки заняты, но победителя нет?

5. На этапе проверки выигрыша

- Сколько всего выигрышных комбинаций существует в игре 3×3?
- Как проверить, что в строке/столбце/диагонали стоят три одинаковых символа?

– Как подсветить выигрышную линию? Какие свойства табличного документа для этого нужны?

– Как определить ничью? Достаточно ли проверить, что все ячейки заполнены?

6. На этапе работы с интерфейсом

– Какие элементы управления нужны на форме? (кнопки, переключатели и т.д.)

– Как сделать так, чтобы кнопка «Играть» была недоступна до выбора символа?

– Когда должна появляться кнопка «Очистить поле»?

– Как сообщить пользователю об ошибке (например, клик по занятой клетке)?

7. На этапе рефлексии и расширения

– Как добавить счётчик побед?

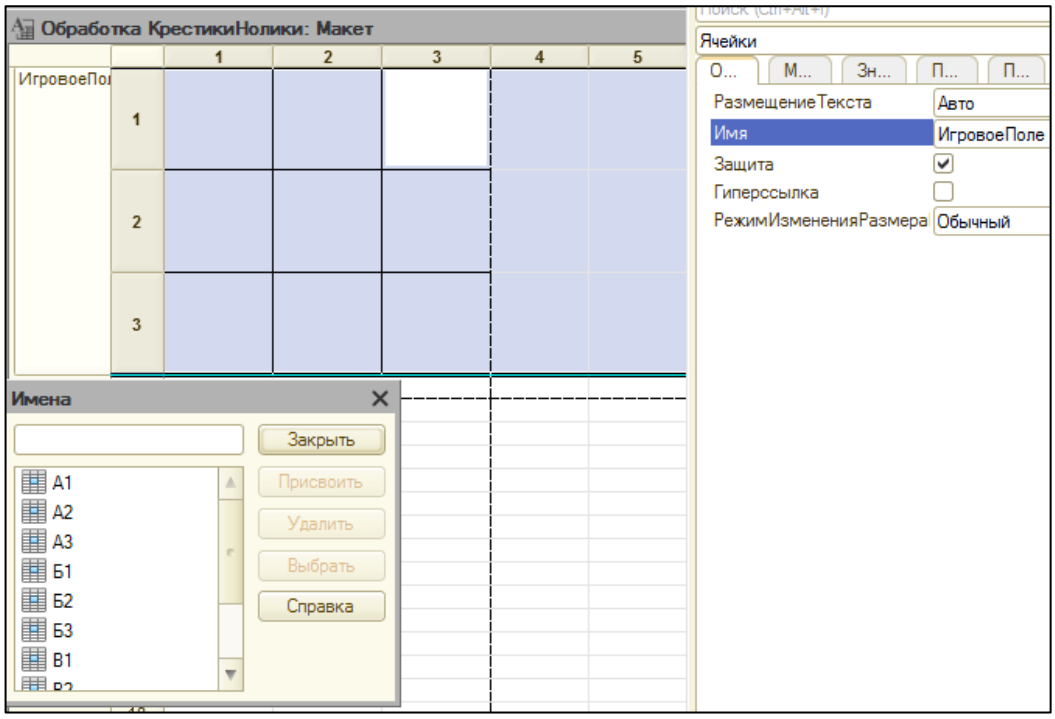
– Можно ли сделать режим «игрок против игрока»? Что для этого изменить?

– Как сохранить историю игр? Какие объекты конфигурации можно использовать?

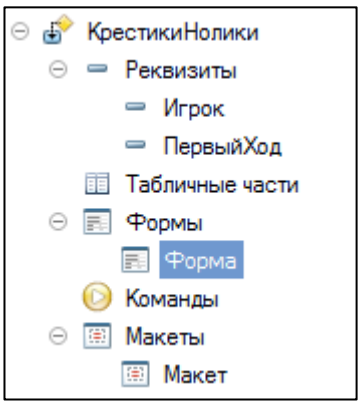
– Можно ли найти чемпиона по результатам игр?

Практическая работа

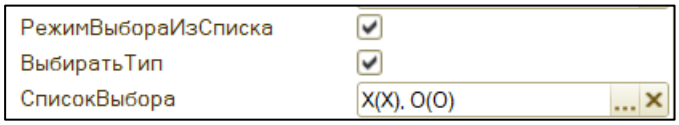
1. Создайте обработку КрестикиНолики.
2. В обработке добавьте Макет – это будет наше игровое поле.
3. Откорректируйте размеры ячеек. Создайте область ИгровоеПоле (Таблица, Имена, Назначить имя).
4. Обратите внимание! Столбцы должны быть названы русскими буквами. Для проверки нужно щёлкнуть левой кнопкой мыши над областью ИгровоеПоле.

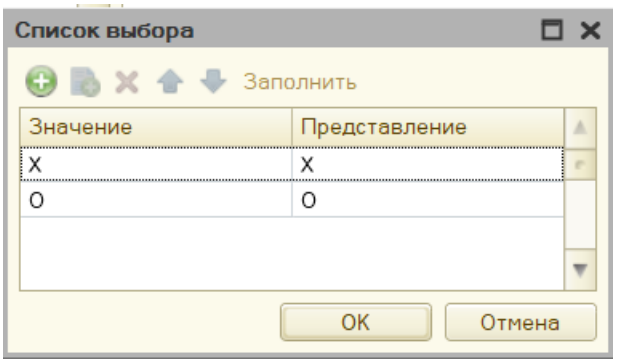


5. Создайте реквизиты обработки: строковый для имени игрока и булевый для указания, будет делать первый ход.



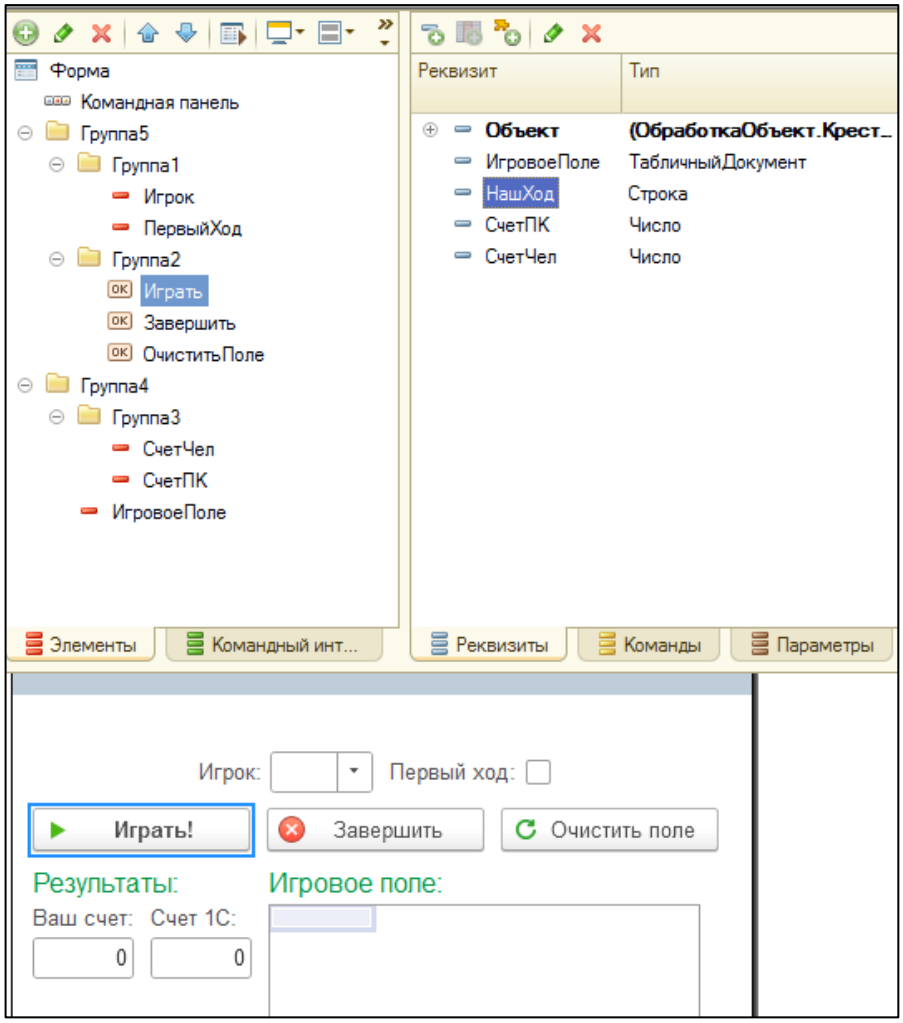
6. Создайте форму и реквизиты для указания счета побед и выбора, какой знак, крестик или нолик, будет использовать игрок. Для размещения игрового поля будет использоваться реквизит с типом ТабличныйДокумент.
7. Для реквизита Игрок включите настройку РежимВыбораИзСписка и настройте список выбора.





8. Создайте кнопки Играть, Завершить, ОчиститьПоле.

9. Оформите интерфейс по своему выбору, используя ОбычныеГруппы, горизонтальное или вертикальное расположение.



Программирование

1. ПриСозданииНаСервере (&НаСервере), процедура для формы.

Назначение: Инициализация формы при её открытии — загрузка игрового поля из макета.

Алгоритм:

1. Получает ссылку на саму обработку через РеквизитФормыВЗначение("Объект").
2. Загружает макет с именем "Макет".
3. Извлекает из макета именованную область "ИгровоеПоле".
4. Выводит эту область в табличный документ на форме.

5. Блокирует игровое поле (Доступность = Ложь), так как игра ещё не начата.
6. Скрывает кнопку «Очистить поле» (Видимость = Ложь).

Выполняется на сервере, потому что работа с макетами возможна только там.

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    // Получаем ссылку на текущую внешнюю обработку
    ОбъектОбработка = РеквизитФормыВЗначение("Объект");
    // Загружаем макет с именем "Макет"
    Макет = ОбъектОбработка.ПолучитьМакет("Макет");
    // Извлекаем из макета именованную область "ИгровоеПоле"
    ИгровоеПоле = Макет.ПолучитьОбласть("ИгровоеПоле");
    // Выводим содержимое области в табличный документ на форме
    Макет.Вывести(ИгровоеПоле);
    // Блокируем игровое поле — игра ещё не начата
    Элементы.ИгровоеПоле.Доступность = Ложь;
    // Скрываем кнопку "Очистить поле" — она появится только
после завершения игры
    Элементы.ОчиститьПоле.Видимость = Ложь;
КонецПроцедуры
```

2. Играть (&НаКлиенте)

Назначение: Запуск игры после выбора символа игроком.

Алгоритм:

1. Проверяет, выбран ли символ (Объект.Игрок не пуст). Если нет — показывает сообщение об ошибке и завершает выполнение.
2. Определяет символ компьютера. Если игрок выбрал "X", то компьютер играет "O", и наоборот.
3. Блокирует кнопку «Играть» (чтобы нельзя было запустить игру дважды).
4. Проверяет, ходит ли **компьютер первым** (Объект.ПервыйХод = Ложь):

- Если да — ставит свой символ в центр (Б2 или Область(2,2)).
- Если нет — выводит подсказку: «Ваш ход!».

5. Разблокирует игровое поле для ходов игрока.

Эта процедура запускает игру и определяет, кто делает первый ход.

```
&НаКлиенте
Процедура Играть(Команда)
// Обработчик нажатия кнопки "Играть"
&НаКлиенте
Процедура Играть(Команда)
     // Проверяем, выбрал ли игрок символ ("X" или "O")
     Если Объект.Игрок = "" Тогда
          Сообщение = Новый СообщениеПользователю;
          Сообщение.Текст = "Выберите крестик или нолик!";
          Сообщение.Поле = "Объект.Игрок"; // Подсвечивает поле
выбора
          Сообщение.Сообщить();
          Возврат; // Прерываем запуск игры
     ИначеЕсли
          Объект.Игрок = "X" Тогда
          НашХод = "O"; // Компьютер играет "O"
     Иначе
          НашХод = "X"; // Компьютер играет "X"
     КонецЕсли;
     // Блокируем кнопку "Играть", чтобы нельзя было запустить
игру дважды
     Элементы.Играть.Доступность = Ложь;
     // Если игрок НЕ ходит первым — компьютер делает первый ход
в центр (Б2)
     Если Объект.ПервыйХод = Ложь Тогда
          ИгровоеПоле.Область(2, 2).Текст = НашХод; // Центр поля
— 2-я строка, 2-й столбец
     Иначе
```

```
          ПоказатьЗначение(, "Ваш ход!"); // Подсказка игроку
     КонецЕсли;
     // Разблокируем игровое поле для ходов игрока
     Элементы.ИгровоеПоле.Доступность = Истина;
КонецПроцедуры
```

3. ИгровоеПолеВыбор (&НаКлиенте)

Назначение: Обработка клика игрока по ячейке.

Алгоритм:

1. Проверяет, свободна ли выбранная ячейка (Область.Текст = "").
2. Если свободна:
   - ставит в неё символ игрока (Объект.Игрок);
   - блокирует поле (чтобы нельзя было кликнуть снова до хода компьютера);
   - вызывает СделатьСледующийХод() — ход компьютера.
3. Если занята — выводит сообщение: «Уже занято!».

Это точка входа в игровой цикл после действия пользователя.

```
// Обработчик клика по ячейке игрового поля
&НаКлиенте
Процедура ИгровоеПолеВыбор(Элемент, Область,
СтандартнаяОбработка)
     // Проверяем, свободна ли выбранная ячейка
     Если Область.Текст = "" Тогда
          // Ставим символ игрока в ячейку
          Область.Текст = Объект.Игрок;
          // Блокируем поле до завершения хода компьютера
          Элементы.ИгровоеПоле.Доступность = Ложь;
          // Передаём ход компьютеру
          СделатьСледующийХод();
     Иначе
          // Если ячейка занята — выводим предупреждение
          ПоказатьЗначение(, "Уже занято!");
```

```
          Возврат;
     КонецЕсли;
КонецПроцедуры
```

4. СделатьСледующийХод (&НаКлиенте)

Назначение: реализует логику хода компьютера.

Алгоритм:

1. Проверяет, не закончилась ли игра после хода игрока → если да, завершает.
2. Попытка выиграть: вызывает ПроверитьПозиции(..., НашХод). Если найдена линия из двух своих символов — ставит третий и выигрывает.
3. Если не может выиграть — блокирует угрозу: вызывает ПроверитьПозиции(..., Объект.Игрок). Если игрок может выиграть следующим ходом — компьютер блокирует его.
4. Если нет угроз — занимает центр (Б2), если он свободен.
5. Если центр занят — занимает первую свободную ячейку (по строкам сверху вниз, слева направо).
6. После хода снова проверяет, не закончилась ли игра.
7. Если игра продолжается — разблокирует поле для следующего хода игрока.
8. Если игра окончена — блокирует поле и показывает кнопку «Очистить поле».

Это основной ИИ компьютера — простой, но эффективный.

```
// Реализует ход компьютера (искусственный интеллект)
&НаКлиенте
Процедура СделатьСледующийХод()
     ХодСделан = Ложь;
     // Проверяем, не закончилась ли игра после хода игрока
     Соответствие = ЗаполнитьСоответствие();
     КонецИгры = ПроверитьВозможностьВыигрыша(Соответствие);
     Если КонецИгры = Истина Тогда
          Элементы.ИгровоеПоле.Доступность = Ложь;
```

```
          Элементы.ОчиститьПоле.Видимость = Истина;
          Возврат;
     КонецЕсли;
     // 1. Пытаемся ВЫИГРАТЬ: ищем линию, где уже два наших
символа
     ХодСделан = ПроверитьПозиции(Соответствие, НашХод);
     // 2. Если не можем выиграть — БЛОКИРУЕМ угрозу: линия с
двумя символами игрока
     Если ХодСделан = Ложь Тогда
          ХодСделан = ПроверитьПозиции(Соответствие,
Объект.Игрок);
     КонецЕсли;
     // 3. Если нет угроз — занимаем ЦЕНТР (Б2), если он свободен
     Если ХодСделан = Ложь И ИгровоеПоле.Области.Б2.Текст = ""
Тогда
         ИгровоеПоле.Области.Б2.Текст = НашХод;
          ХодСделан = Истина;
     КонецЕсли;
     // 4. Если центр занят — занимаем первую свободную ячейку
(слева-направо, сверху-вниз)
     Если ХодСделан = Ложь Тогда
          Для НомерСтроки = 1 По 3 Цикл
               Для НомерСтолбца = 1 По 3 Цикл
               Если ИгровоеПоле.Область(НомерСтроки,
НомерСтолбца).Текст = "" Тогда
               ИгровоеПоле.Область(НомерСтроки,
НомерСтолбца).Текст = НашХод;
                         ХодСделан = Истина;
                         Прервать;
                    КонецЕсли;
               КонецЦикла;
               Если ХодСделан = Истина Тогда
                   Прервать;
               КонецЕсли;
          КонецЦикла;
     КонецЕсли;
```

```
    // После хода компьютера снова проверяем завершение игры
    Соответствие = ЗаполнитьСоответствие();
    КонецИгры = ПроверитьВозможностьВыигрыша(Соответствие);
    Если КонецИгры = Ложь Тогда
        // Игра продолжается — разблокируем поле для следующего
хода игрока
        Элементы.ИгровоеПоле.Доступность = Истина;
    Иначе
        // Игра окончена — блокируем поле и показываем кнопку
"Очистить поле"
        Элементы.ИгровоеПоле.Доступность = Ложь;
        Элементы.ОчиститьПоле.Видимость = Истина;
    КонецЕсли;
КонецПроцедуры
```

5. ПроверитьПозиции **(**&НаКлиенте**)**

Назначение: проверяет, есть ли в какой-либо линии **две одинаковые фигуры** указанного типа (Ход).

Алгоритм:

1. Проходит по всем линиям (строки, столбцы, диагонали) из Соответствие.
2. Для каждой линии собирает тексты ячеек в строку (например: "XX").
3. Если строка длиной 2 и состоит из двух одинаковых символов (Ход + Ход):
   - находит **пустую ячейку** в этой линии;
   - ставит туда **свой символ** (НашХод);
   - возвращает Истина — ход сделан.
4. Если ни одна линия не подошла — возвращает Ложь.

Используется дважды: сначала для атаки, потом для защиты.

```
// Проверяет, есть ли в какой-либо линии ДВА одинаковых символа
указанного типа (Ход)
// Если да — ставит свой символ в пустую ячейку этой линии
&НаКлиенте
```

```
Функция ПроверитьПозиции(Соответствие, Ход)
     ХодСделан = Ложь;
     // Проходим по всем линиям
     Для каждого ЭлементСоответствия Из Соответствие Цикл
          МассивТекст = Новый Массив;
          // Собираем тексты ячеек
     Для каждого Элемент Из ЭлементСоответствия.Значение Цикл
               МассивТекст.Добавить(Элемент.Текст);
          КонецЦикла;
          СтрокаКомбинации = СтрСоединить(МассивТекст);
     // Если в линии ровно два символа и они совпадают с "Ход"
          Если СтрДлина(СтрокаКомбинации) = 2 И СтрокаКомбинации
= Ход + Ход Тогда
     // Получаем массив ячеек этой линии
               КомбинацияЯчеек =
Соответствие.Получить(ЭлементСоответствия.Ключ);
     // Ищем пустую ячейку и ставим туда СВОЙ символ (НашХод)
               Для каждого ТекЯчейка Из КомбинацияЯчеек Цикл
                    Если ТекЯчейка.Текст = "" Тогда
                        ТекЯчейка.Текст = НашХод;

                         ХодСделан = Истина;
                         Прервать;
                    КонецЕсли;
               КонецЦикла;
               Если ХодСделан = Истина Тогда
                    Прервать;
               КонецЕсли;
          КонецЕсли;
     КонецЦикла;
     Возврат ХодСделан; // Возвращает Истина, если ход был сделан
КонецФункции
```

6. ЗаполнитьСоответствие **(**&НаКлиенте**)**

Назначение: формирует структуру всех 8 выигрышных комбинаций.

Алгоритм:

Создаёт Соответствие, где:

- **ключи** — названия линий: "ПерваяСтрока", "Диагональ1" и т.д.
- **значения** — массивы из 3 ссылок на ячейки табличного документа.

Например: [ИгровоеПоле.Области.А1, А2, А3] — первая строка.

```
// Формирует структуру всех 8 возможных выигрышных линий (3 строки, 3 столбца, 2 диагонали)
&НаКлиенте
Функция ЗаполнитьСоответствие()
    Соответствие = Новый Соответствие;
    // Диагональ 1: А1 → Б2 → В3
    МассивДиагональ1 = Новый Массив;
    МассивДиагональ1.Добавить(ИгровоеПоле.Области.А1);
    МассивДиагональ1.Добавить(ИгровоеПоле.Области.Б2);
    МассивДиагональ1.Добавить(ИгровоеПоле.Области.В3);
    Соответствие.Вставить("Диагональ1", МассивДиагональ1);
    // Диагональ 2: В1 → Б2 → А3
    МассивДиагональ2 = Новый Массив;
    МассивДиагональ2.Добавить(ИгровоеПоле.Области.В1);
    МассивДиагональ2.Добавить(ИгровоеПоле.Области.Б2);
    МассивДиагональ2.Добавить(ИгровоеПоле.Области.А3);
    Соответствие.Вставить("Диагональ2", МассивДиагональ2);
    // Строки
    МассивПерваяСтрока = Новый Массив;
    МассивПерваяСтрока.Добавить(ИгровоеПоле.Области.А1);
    МассивПерваяСтрока.Добавить(ИгровоеПоле.Области.А2);
    МассивПерваяСтрока.Добавить(ИгровоеПоле.Области.А3);
    Соответствие.Вставить("ПерваяСтрока", МассивПерваяСтрока);
    МассивВтораяСтрока = Новый Массив;
    МассивВтораяСтрока.Добавить(ИгровоеПоле.Области.Б1);
    МассивВтораяСтрока.Добавить(ИгровоеПоле.Области.Б2);
    МассивВтораяСтрока.Добавить(ИгровоеПоле.Области.Б3);
```

```
    Соответствие.Вставить("ВтораяСтрока", МассивВтораяСтрока);
     МассивТретьяСтрока = Новый Массив;
     МассивТретьяСтрока.Добавить(ИгровоеПоле.Области.В1);
     МассивТретьяСтрока.Добавить(ИгровоеПоле.Области.В2);
     МассивТретьяСтрока.Добавить(ИгровоеПоле.Области.В3);
     Соответствие.Вставить("ТретьяСтрока", МассивТретьяСтрока);
     // Столбцы
     МассивПервыйСтолбец = Новый Массив;
     МассивПервыйСтолбец.Добавить(ИгровоеПоле.Области.А1);
     МассивПервыйСтолбец.Добавить(ИгровоеПоле.Области.Б1);
     МассивПервыйСтолбец.Добавить(ИгровоеПоле.Области.В1);
     Соответствие.Вставить("ПервыйСтолбец", МассивПервыйСтолбец);
    МассивВторойСтолбец = Новый Массив;
     МассивВторойСтолбец.Добавить(ИгровоеПоле.Области.А2);
     МассивВторойСтолбец.Добавить(ИгровоеПоле.Области.Б2);
     МассивВторойСтолбец.Добавить(ИгровоеПоле.Области.В2);
     Соответствие.Вставить("ВторойСтолбец", МассивВторойСтолбец);
    МассивТретийСтолбец = Новый Массив;
     МассивТретийСтолбец.Добавить(ИгровоеПоле.Области.А3);
     МассивТретийСтолбец.Добавить(ИгровоеПоле.Области.Б3);
     МассивТретийСтолбец.Добавить(ИгровоеПоле.Области.В3);
     Соответствие.Вставить("ТретийСтолбец", МассивТретийСтолбец);
     Возврат Соответствие; // Возвращает структуру всех линий
КонецФункции
```

7. ПроверитьВозможностьВыигрыша **(**&НаКлиенте**)**

Назначение: проверяет, есть ли победитель или ничья.

Алгоритм:

1. Формирует строки-выигрыши: "XXX" и "OOO".
2. Проходит по всем линиям из Соответствие:

– собирает тексты ячеек в строку;

– если строка = "XXX" или "OOO":

- вызывает ОбработкаВыигрыша(...);
- возвращает Истина (игра окончена).

3. Если победителя нет, проверяет **заполненность поля**: если все 3 строки заполнены (не пустые) → объявляет **ничью.**
4. Возвращает Истина, если игра окончена, иначе — Ложь.

Главная функция определения конца игры.

```
// Проверяет, завершилась ли игра (победа или ничья)
&НаКлиенте
Функция ПроверитьВозможностьВыигрыша(Соответствие)
     КонецИгры = Ложь;
     // Формируем выигрышные строки для компьютера и игрока
     НашВыигрыш = НашХод + НашХод + НашХод;        // Например: "ООО"
     ЧужойВыигрыш = Объект.Игрок + Объект.Игрок + Объект.Игрок; // Например: "XXX"
     // Для проверки ничьи будем запоминать содержимое строк
     ПерваяСтрока = "";
     ВтораяСтрока = "";
     ТретьяСтрока = "";
     // Проходим по всем линиям (строки, столбцы, диагонали)
     Для каждого ЭлементСоответствия Из Соответствие Цикл
          МассивТекст = Новый Массив;
          // Собираем тексты всех ячеек в текущей линии
     Для каждого Элемент Из ЭлементСоответствия.Значение Цикл
               МассивТекст.Добавить(Элемент.Текст);
          КонецЦикла;
          // Объединяем в одну строку (например: "XOX")
          СтрокаКомбинации = СтрСоединить(МассивТекст);

// Рассматриваем только полностью заполненные линии (длина = 3)
          Если СтрДлина(СтрокаКомбинации) = 3 Тогда
               // Запоминаем содержимое строк для проверки ничьей
          Если ЭлементСоответствия.Ключ = "ПерваяСтрока" Тогда
                    ПерваяСтрока = СтрокаКомбинации;
```

```
                КонецЕсли;
            Если ЭлементСоответствия.Ключ = "ВтораяСтрока" Тогда
                ВтораяСтрока = СтрокаКомбинации;
                КонецЕсли;
            Если ЭлементСоответствия.Ключ = "ТретьяСтрока" Тогда
                ТретьяСтрока = СтрокаКомбинации;
                КонецЕсли;
                // Проверяем, есть ли выигрышная комбинация
                Если СтрокаКомбинации = НашВыигрыш Тогда
                // Компьютер выиграл
    ОбработкаВыигрыша(ЭлементСоответствия.Значение, Истина);
                 КонецИгры = Истина;
                Прервать; // Выходим из цикла
                ИначеЕсли СтрокаКомбинации = ЧужойВыигрыш Тогда
                // Игрок выиграл
    ОбработкаВыигрыша(ЭлементСоответствия.Значение, Ложь);
                 КонецИгры = Истина;
                Прервать;
                КонецЕсли;
            КонецЕсли;
    КонецЦикла;
    // Если победителя нет, проверяем ничью: все строки
заполнены
    Если КонецИгры = Ложь Тогда
            Если ПерваяСтрока <> "" И ВтораяСтрока <> "" И
ТретьяСтрока <> "" Тогда
                ПоказатьЗначение(, "Ничья!");
                КонецИгры = Истина;
            КонецЕсли;
    КонецЕсли;
    Возврат КонецИгры; // Возвращает Истина, если игра окончена
КонецФункции
```

8. ОбработкаВыигрыша (&НаКлиенте)

Назначение: Визуальное оформление победы/поражения.

Алгоритм:

1. Определяет, кто выиграл:
   - если НашВыигрыш = Ложь → выиграл **игрок** → увеличивает СчетЧел;
   - иначе → выиграл **компьютер** → увеличивает СчетПК.
2. Подсвечивает **все ячейки выигрышной линии** цветом WebЦвета.ЛососьСветлый.
3. Выводит сообщение: «Игра окончена. Вы выиграли!» или «...проиграли!».

Отвечает за визуальную обратную связь.

```
// Визуальная обработка завершения игры (подсветка, счёт,
сообщение)
&НаКлиенте
Процедура ОбработкаВыигрыша(СтрокаВыигрыша, НашВыигрыш)
     // Определяем, кто выиграл, и обновляем счёт
     Если НашВыигрыш = Ложь Тогда
         Итог = "выиграли!";   // Игрок выиграл
          СчетЧел = СчетЧел + 1;
     Иначе
          Итог = "проиграли!";  // Компьютер выиграл
          СчетПК = СчетПК + 1;
     КонецЕсли;
     // Подсвечиваем выигрышную линию цветом
     Для каждого Элемент Из СтрокаВыигрыша Цикл
          Элемент.ЦветФона = WebЦвета.ЛососьСветлый;
     КонецЦикла;
     // Выводим итоговое сообщение
     ПоказатьЗначение(, "Игра окончена. Вы " + Итог);
КонецПроцедуры
```

9. ОбновитьЭлементыФормы (&НаКлиенте)

Назначение: Сброс игрового поля к начальному состоянию.

Алгоритм:

1. Очищает весь диапазон A1:C3 (через Область(1,1,3,3).Текст = "").
2. Сбрасывает фон ячеек на белый.
3. Разблокирует кнопку «Играть».
4. Блокирует игровое поле.
5. Скрывает кнопку «Очистить поле».

Используется при перезапуске игры.

```
&НаКлиенте
Процедура ОбновитьЭлементыФормы()
     ИгровоеПоле.Область(1,1,3,3).Текст = "";
     ИгровоеПоле.Область(1,1,3,3).ЦветФона = WebЦвета.Белый;
     Элементы.Играть.Доступность = Истина;
     Элементы.ИгровоеПоле.Доступность = Ложь;
    Элементы.ОчиститьПоле.Видимость = Ложь;
КонецПроцедуры
```

10. ОчиститьПоле (&НаКлиенте)

Назначение: обработчик нажатия кнопки «Очистить поле».

Алгоритм: просто вызывает ОбновитьЭлементыФормы().

```
&НаКлиенте
Процедура ОчиститьПоле(Команда)
     ОбновитьЭлементыФормы();
КонецПроцедуры
```

11. Завершить и Закрытие (&НаКлиенте)

Назначение: Корректное завершение игры с подтверждением.

Алгоритм:

1. Завершить: Показывает диалог: «Вы хотите прекратить игру?». Использует **асинхронное оповещение** → вызывает Закрытие при ответе.

2. Закрытие: Если ответ «Да» → сбрасывает игру через ОбновитьЭлементыФормы().

Позволяет выйти из игры без закрытия всей формы.

```
&НаКлиенте

Процедура Завершить(Команда)

    ОповещениеОЗакрытии = Новый ОписаниеОповещения("Закрытие",
ЭтотОбъект);

    ПоказатьВопрос(ОповещениеОЗакрытии, "Вы хотите прекратить
игру?", РежимДиалогаВопрос.ДаНет);

КонецПроцедуры



&НаКлиенте

Процедура Закрытие(РезультатВопроса, ДополнительныеПараметры)
Экспорт

    Если РезультатВопроса = КодВозвратаДиалога.Да Тогда

        ОбновитьЭлементыФормы();

    Иначе

        Возврат;

    КонецЕсли;

КонецПроцедуры
```

Самостоятельная работа

1. Организуйте вывод на экран сообщения о том, кто является лидером текущей игры.
2. Добавьте поле для ввода имени игрока, записывайте его результат в какой-либо объект конфигурации.
3. Определите чемпиона игры «Крестики-нолики».

Подведение итогов

1. Какие ещё игры можно сделать в 1С, предложите варианты.
2. Распределите предложенные игры по уровню сложности.
3. Какие из игр предполагают программирование действий бота?

Создание игры «Блэкджек» (двадцать одно очко)

Разработана студентом группы ИС-31 Лосевым Артёмом

Цель: разработать карточную игру с графическим интерфейсом и возможность игры с ботом.

1. Блэкджек – правила игры

Блэкджек (или двадцать одно очко) – представляет собой небольшую игру с картами, в которой, естественно, вам необходимо одержать вверх над противником. Цель игры: набрать двадцать одно очко. В начале игры вам и вашему противнику раздаётся по две карты. Карты выпадают случайным образом от 1 до 11. При этом карты в колоде не повторяются. Каждый свой ход вы можете выбрать один из двух вариантов: взять карту или остаться при своих (что означает – пас). После любого принятого вами решения инициатива выбора переходит к противнику, у которого тоже имеется выбор: взять карту или же нет. Так как видеть все карты противника было бы не так интересно, первая карта, которую он получает, помечается не цифрой, а знаком «?», заставляя игрока прикидывать реальное количество очков противника. Игра заканчивается, когда вы и противник ответите «Пас». После будет производится подсчёт очков следующим образом:

− если вы набрали больше очков, чем противник, и при этом не перебрали за 21 очко, то вы одерживаете вверх;

− если же вы набрали больше очков, чем противник, и при этом вы перебрали за 21 очко (к примеру, в сумме у вас уже 23 очка), тогда вы проигрываете;

− если вы и ваш противник перебрали за 21 очко, то выигрывает тот, у кого меньше всего очков;

− если вы и ваш противник набрали по равному количеству очков – это считается ничьёй.

2. План работы

Перед тем как реализовывать игру, необходимо понимать, что конкретно для это необходимо для выполнение поставленной задачи. Разобьём разработку на несколько этапов:

− подготовка изображений и информационной базы 1С;

− реализация формы в обработке;

− реализация логики в модуле формы;

− реализация ИИ противника;

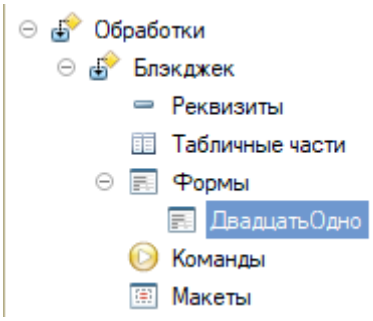
− тестирование.

План представляет из себя пять пунктов и выполнять всё необходимо в строгой последовательности. Также, если вы способны реализовать более грамотные, структурированные и просто оптимальные решения для определённых участков кода или формы, нежели те, что представлены в этом путеводители – смело можете использовать и их.

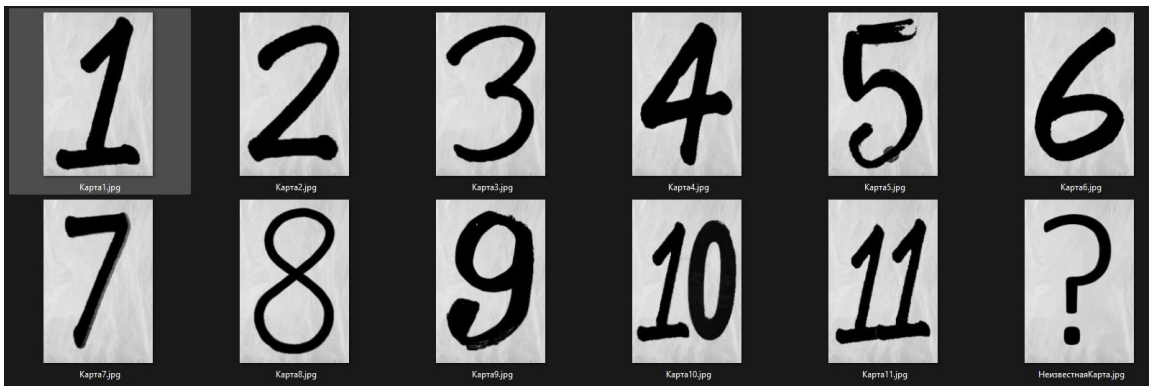
3. Подготовка рабочей базы и ресурсов

Для начала, как и всегда, создадим новую информационную базу (ИБ) в 1С. Можете задать ИБ любое другое имя, которое считаете нужным.

После создания ИБ запускаем конфигуратор и создаём новую обработку. Называем её любым именем или оставляем данное имя программой. После создаём форму. По итогу получается такая структура:

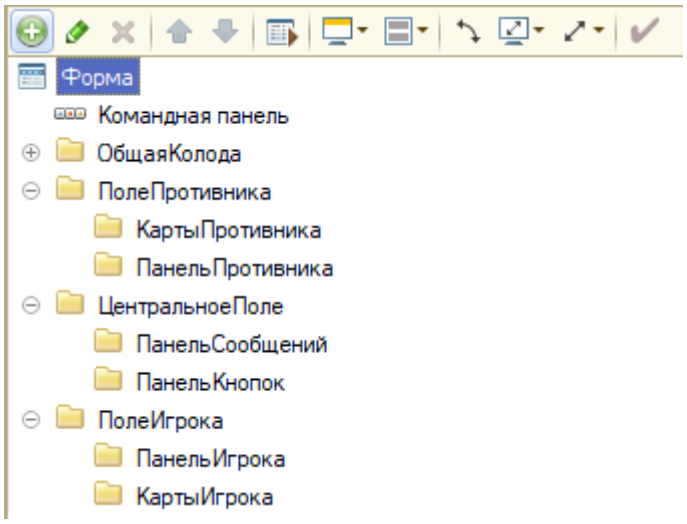


Последней блок в этом пункте – это изображение карт. Их вы можете найти в той же папки, что и эту документацию. Откройте папку «Ресурсы» и убедитесь в том, что там присутствуют все 12 карт, включая вариант неизвестной карты.



4. Реализация формы в обработке

Создайте следующую иерархию папок, используя функцию «Добавить» (белый плюс на зелёном фоне), а после «Группа – обычная группа без отображения». Используйте стрелочки или перетаскивание папок, чтобы составить структуру как на скриншоте:



Далее выбираем группу «Общая колода» и добавляем в нём «Декорация - картинка». Когда добавите её необходимо будет поработать с её свойствами. Ориентируясь по скриншоту (рис. 1), поставьте следующие свойства: изменить имя, как указано на скриншоте, уберите галочки с параметра Видимости и Доступности, добавьте в параметр Картинка изображение с картой 1, поставьте Размер пропорциональный, сделайте Ширину 11, а Высоту 6.

Добавите точно также ещё десять карт и неизвестную карту с помощью «Декорация - картинка». Дополнительно добавьте декорацию с названием “Прозрачность”, у которой будут такие же свойства, только в параметре

Картинка мы вообще ничего не ставим. В зависимости от карты, что находится в параметре Картинка, меняется и название. По итогу у вас должно быть также как и на рисунке 2.

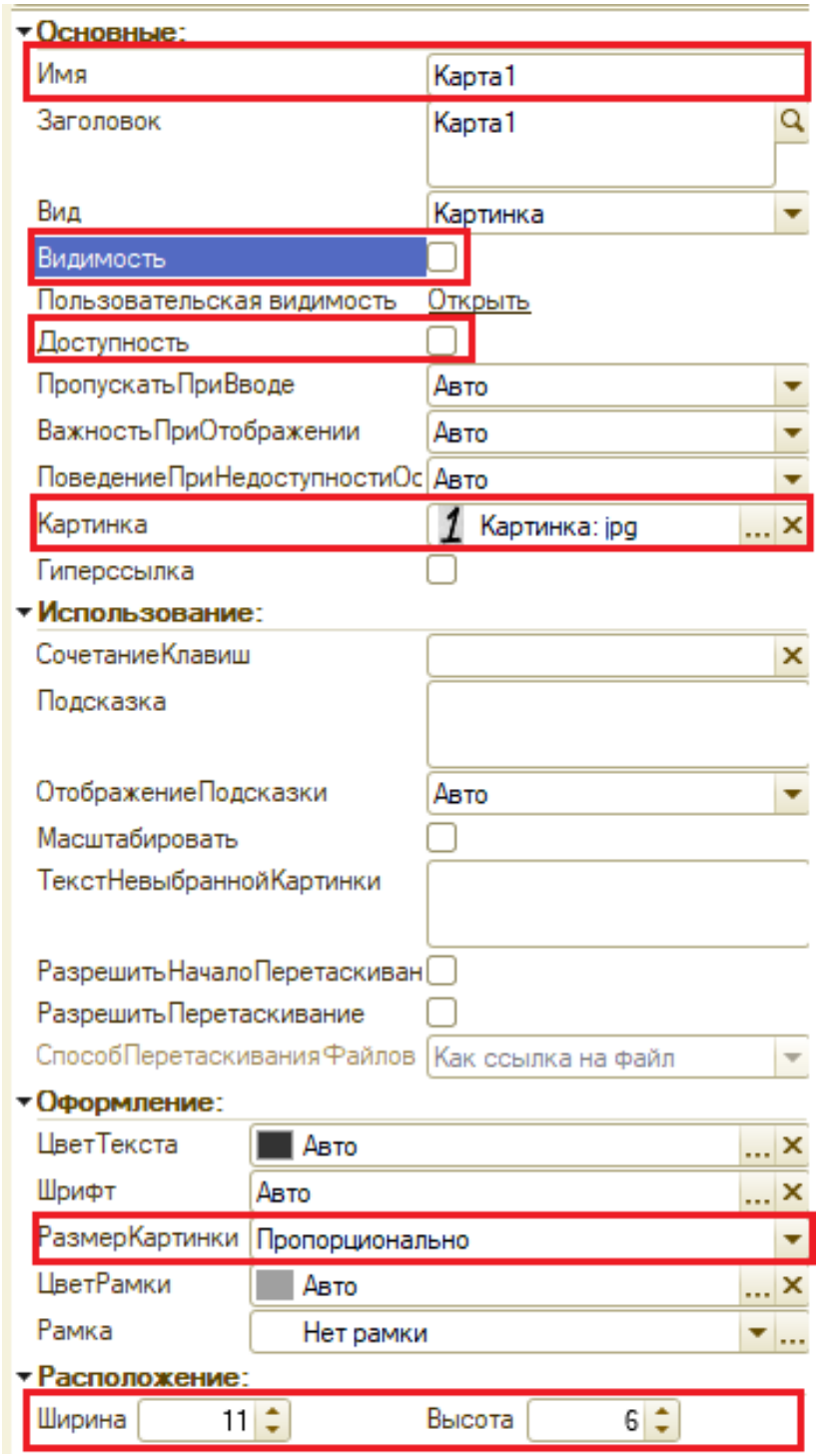


Рисунок 1. Свойства карты

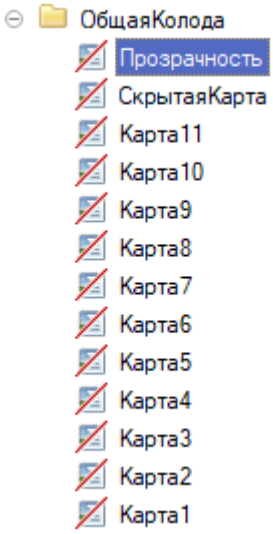


Рисунок 2. Иерархия в группе «ОбщаяКолода»

Задайте Форме следующие свойства, которые указаны на рисунке 3. Также можете в свойствах найти параметр Масштаб и уменьшить его, (допустим до 70) чтобы было лучше видно форму.

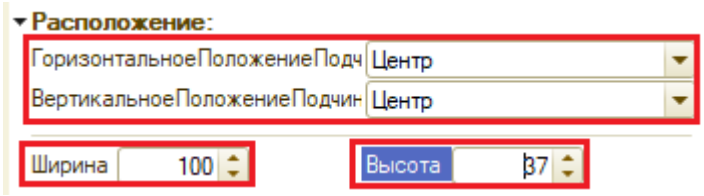


Рисунок 3. Свойства формы

В группу «КартыПротивника» добавьте «Декорация - картинка». На рисунке 3 указано, какие свойства необходимо изменить. Вы можете сами выбрать, каким будет цвет рамки, сама рамка, используемый шрифт и цвет текста. Всего необходимо добавить девять таких слотов. После этого, точно также добавьте девять слотов в группу «КартыИгрока». Не забудьте поставить для группы «КартыПротивника» и «КартыИгрока» в свойствах параметра Группировка «Горизонтальная всегда». Вы также можете изменить цвет группы, в которой находятся слоты. По итогу у вас должен получится результат как на скриншотах 4 и 5.

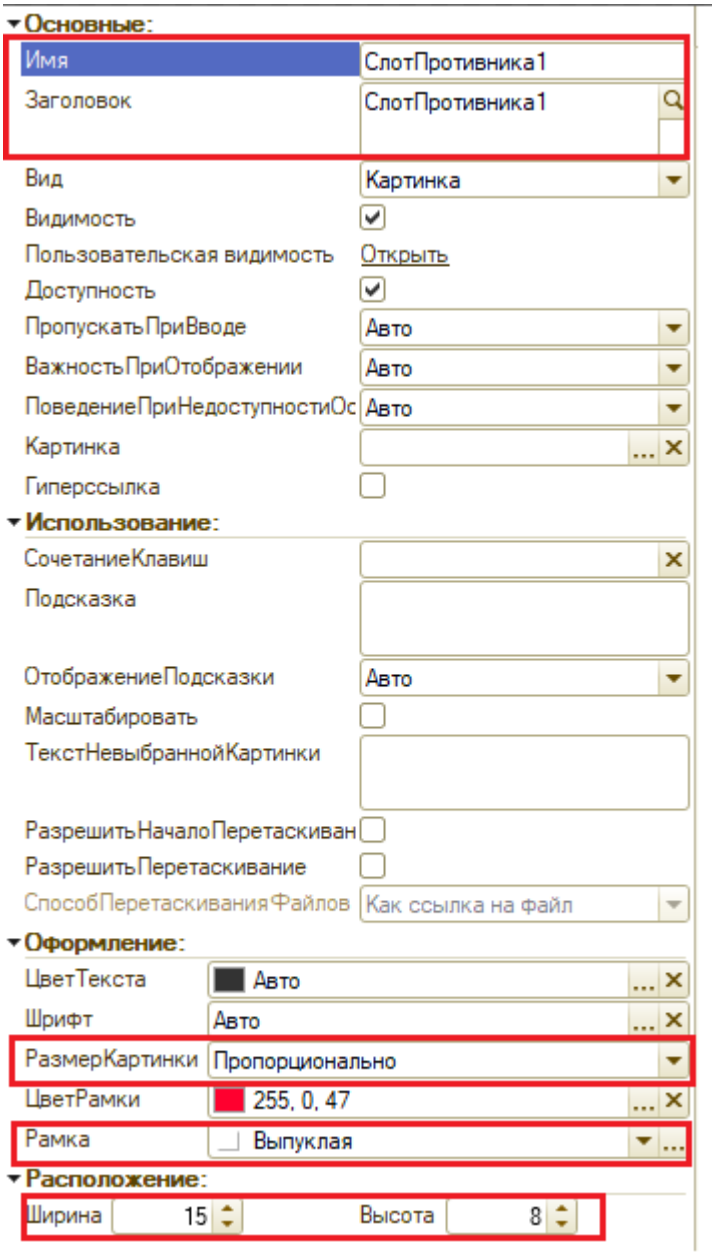


Рисунок 4. Свойства слота

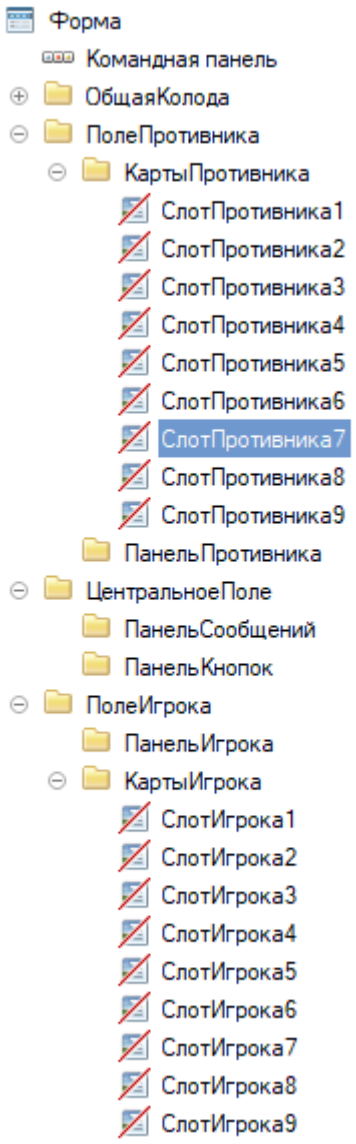


Рисунок 5. Добавленные слоты противнику и игроку

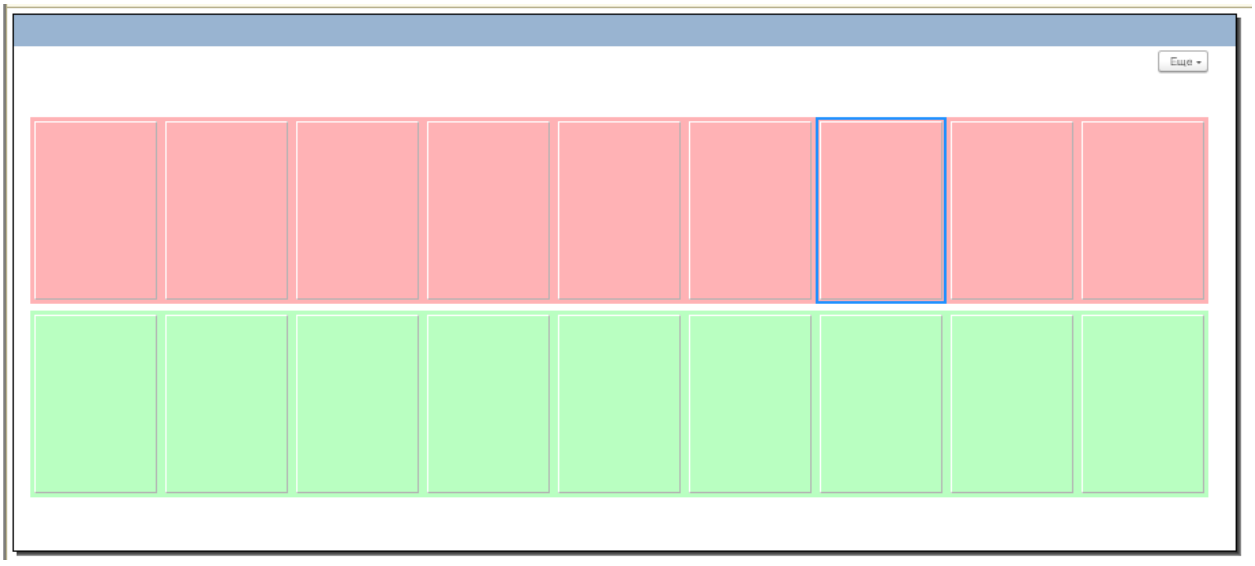
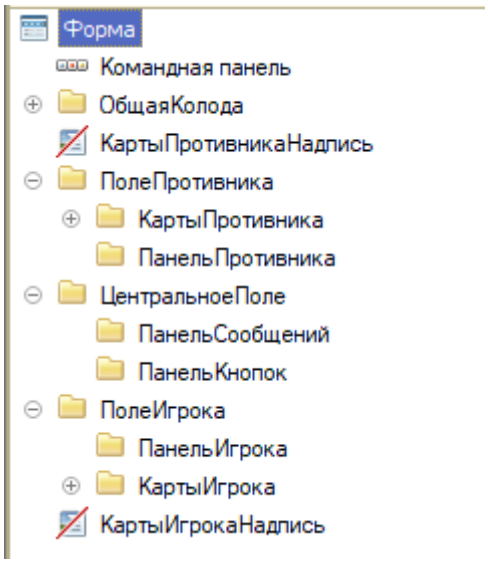
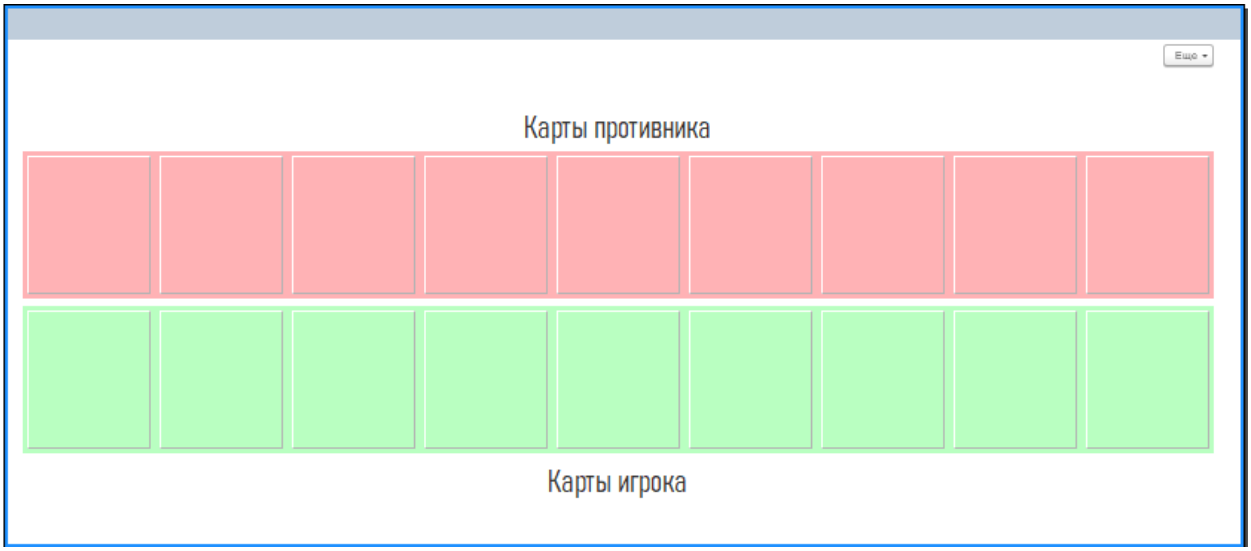


Рисунок 6. Слоты отображаемые на форме

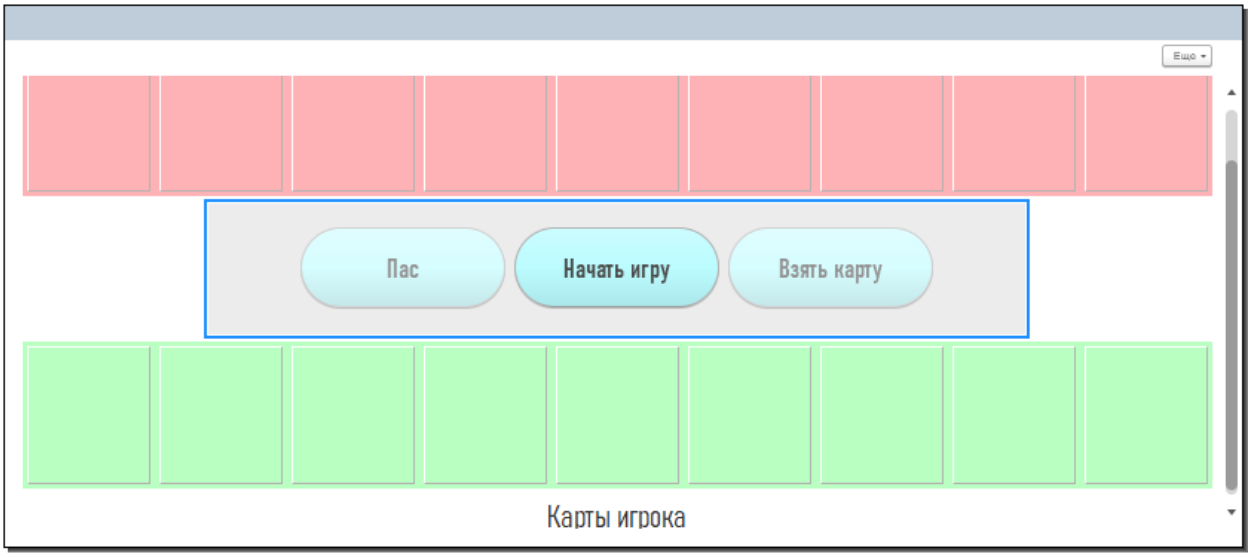
Добавите до группы «КартыПротивника» - «Декорация - надпись». Задайте ей имя «КартыПротивникаНадпись». Также добавьте надпись после группы «КартыИгрока» и задайте ей имя «КартыИгрокаНадпись». Для обоих в заголовки, в Синониме, уберите слово “надпись”. При необходимости измените цвет, шрифт и размер надписи. По итогу у вас должно получится примерно, что изображено на скриншотах:





Выделите группу «ПанельКнопок» в группе «ЦентральноеПоле» и добавьте в неё три кнопки. Чтобы это сделать, зайдите на вкладку «Команды» и добавьте три команды. Задайте командам следующие имена: «Пас», «НачатьИгру» и «ВзятьКарту». Перенесите каждую команду в группу «ЦентральноеПоле». Также не забудьте вновь поставить для группы «ЦентральноеПоле» в

свойствах параметра Группировка «Горизонтальная всегда». Для кнопки «Пас» и «ВзятьКарту» задайте параметр Доступность без галочки. По итогу в вас должно получится:



В группе «ПанельСообщений» создайте «Декорация - надпись». Настройте параметры для этого элемента как показано на рисунке 6. По итогу должно получится как на рисунке 7. Учтите, что оформление может несколько отличатся, в основном, главное правильно назначать имена полям.

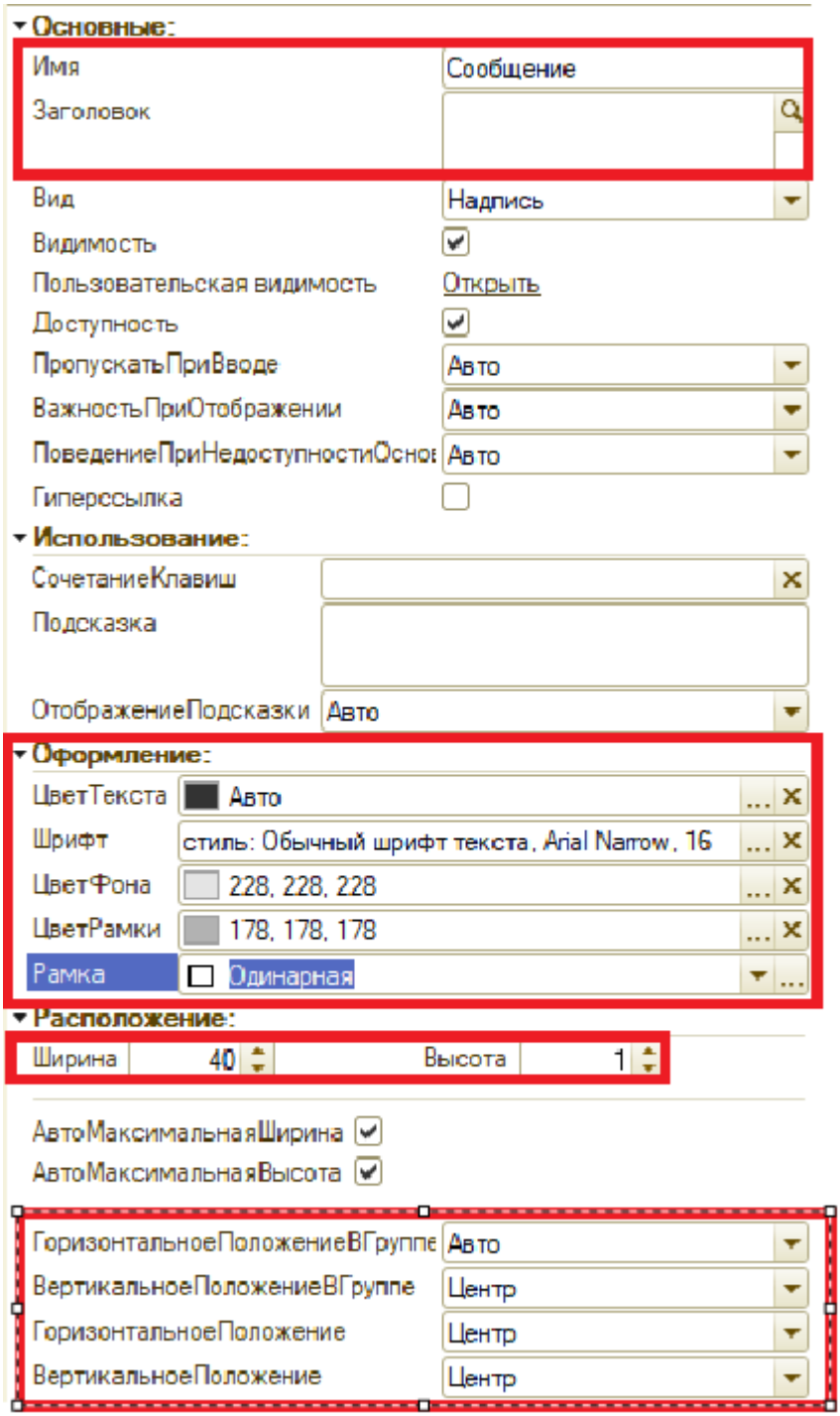
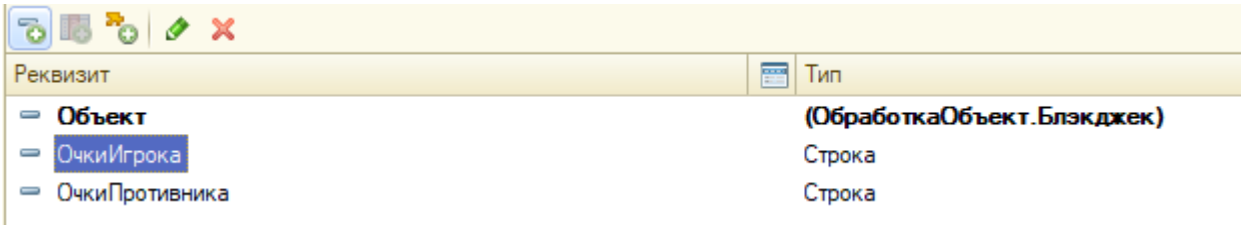


Рисунок 6. Параметры надписи «Сообщение»

Рисунок 7. Добавлен элемент «Сообщение»

Создайте во вкладке Реквизиты два поля «ОчкиИгрока» и «ОчкиПротивника» с типом данных Строка.



Перенесите поле «ОчкиИгрока» в группу «ПанельИгрока», а поле «ОчкиПротивника» в группу «ПанельПротивника» и задайте им параметр Только просмотр на Истина (поставьте галочку). Итого результат на скриншотах (рис 8. и рис. 9). Оформление для полей и их групп задайте сами.

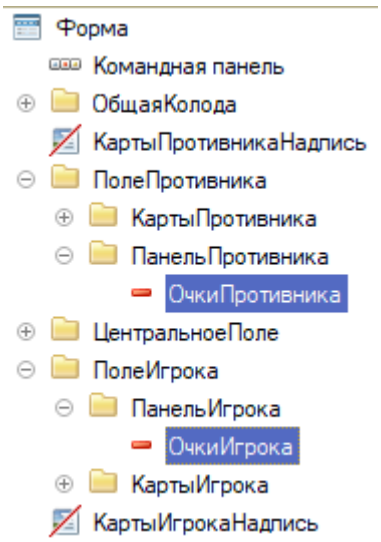


Рисунок 8. Текущая иерархия

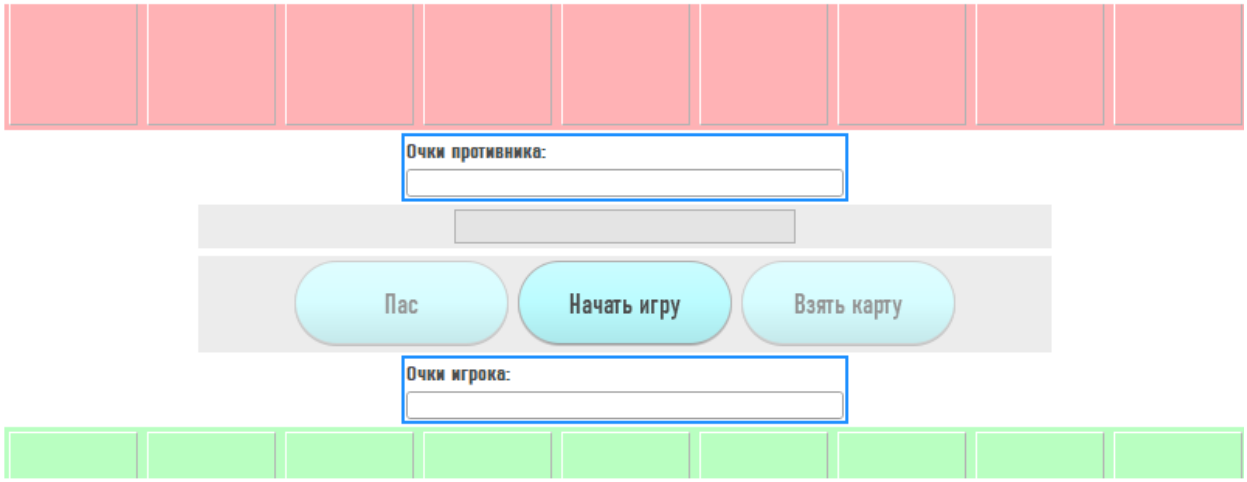
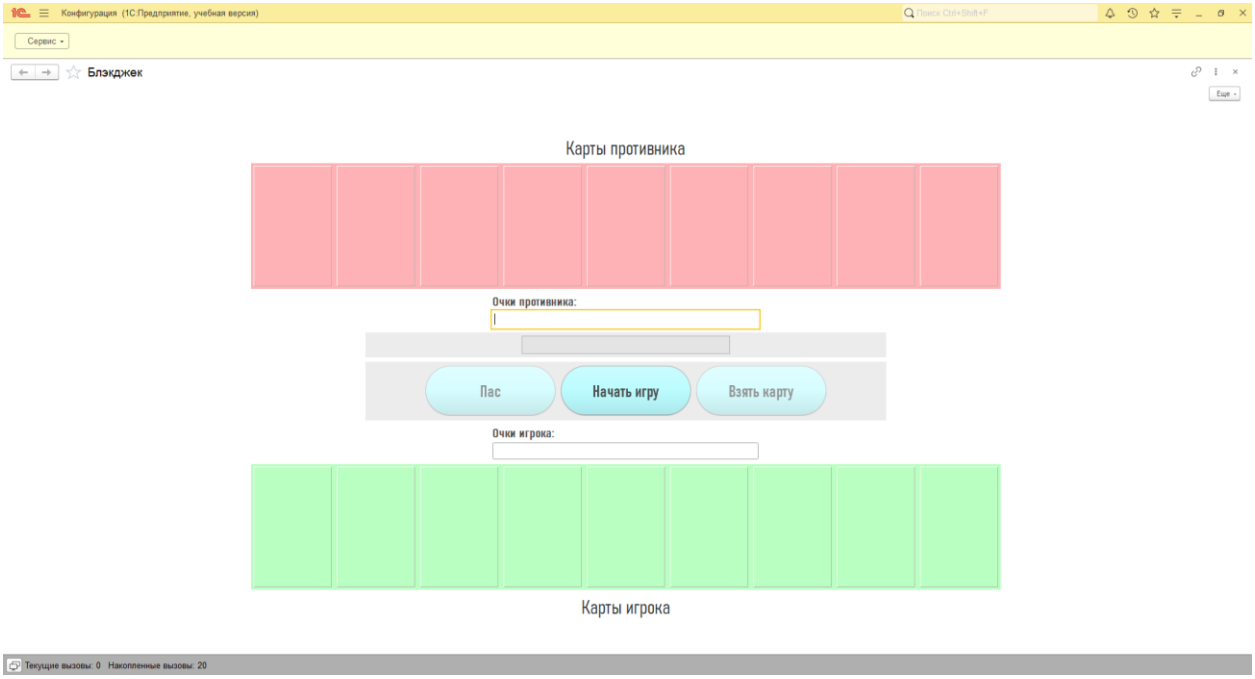


Рисунок 9. Оформление на форме

Отлично! Мы полностью реализовали форму и сделали визуальную составляющую более-менее привлекательной глазу (конечно, если же вы цените оформление :D). На этом второй пункт считается завершённым. Если вы уверены в том, что сделали всё правильно, то можете смело переходить к следующему пункту, после того как посмотрите, что у вас получилось в общем в конфигурации. Если всё кажется маленьким, то увеличьте масштаб формы, если, наоборот, то уменьшите масштаб. Итоговой результат на скриншоте:



5. Реализация логики в модуле обработке

Теперь перейдём в модуль обработки. Пока что он пустой, но это пока что. Исправим это и добавим заранее заготовленные переменные, которые пригодятся нам в будущем. Вставьте этот код в начале обработки:

```
    &НаКлиенте
    Перем Колода, ГлобальныеОчкиПротивника,
ГлобальныеОчкиИгрока, Генератор, СледСлотПротивника,
    СледСлотИгрока, ТекущаяКарта,
СкрытаяКартаПротивника, ОпределениеХода,
    МассивИгрока, МассивПротивника,
АбстрактныйОбщийСчётЗначений,
    АбстрактныйСчётПротивника, АбстрактныйСчётИгрока,
    АнализУспехаПротивника, АнализУспехаИгрока,
    ПасИгрока, ПасПротивника, ДопПрибавка,
ПобедПротивника, ПобедИгрока;
```

Что же значат эти переменные? В принципе, по их именованию можно догадаться. Где-то мы храним колоду и карты на руках противника и игрока. Где-то мы запоминаем определённую карту, запоминаем чей сейчас ход и т.д.

Последние переменные в основном предназначены для обработки входных данных логики ИИ.

Выберите форму кнопки «Начать игру» (ПКМ) - «<Действие команды>» и создайте процедуру на сервере.

```
&НаКлиенте
Перем Колода, ГлобальныеОчкиПротивника, ГлобальныеОчкиИгрока, Генератор, СледСлотПротивника,
СледСлотИгрока, ТекущаяКарта, СкрытаяКартаПротивника, ОпределениеХода,
МассивИгрока, МассивПротивника, АбстрактныйОбщийСчётЗначений,
АбстрактныйСчётПротивника, АбстрактныйСчётИгрока,
АнализУспехаПротивника, АнализУспехаИгрока,
ПасИгрока, ПасПротивника, ДопПрибавка;

&НаКлиенте
Процедура НачатьИгру(Команда)
    // Вставить содержимое обработчика.
КонецПроцедуры
```

Вставьте следующий код в процедуру НачатьИгру:

```
    СледСлотПротивника = 1; СледСлотИгрока = 1;
    ГлобальныеОчкиПротивника = 0;
    ГлобальныеОчкиИгрока = 0;
    АбстрактныйСчётПротивника = 0;
АбстрактныйСчётИгрока = 0;
    АнализУспехаПротивника = 0;
    АнализУспехаИгрока = 0;
    ПобедИгрока = 0; ПобедПротивника = 0;
    ДопПрибавка = 0;
    Генератор = Новый ГенераторСлучайныхЧисел();
    Колода = Новый Массив();
    МассивИгрока = Новый Массив();
    МассивПротивника = Новый Массив();
    АбстрактныйОбщийСчётЗначений = Новый Массив();
    ПасИгрока = Ложь; ПасПротивника = Ложь;
    Для Индекс = 1 По 9 Цикл
        Элементы["СлотПротивника" + Индекс].Картинка =
Элементы.Прозрачность.Картинка;
        Элементы["СлотИгрока" + Индекс].Картинка =
Элементы.Прозрачность.Картинка;
```

```
    КонецЦикла;

    Для Индекс = 1 По 11 Цикл
        Колода.Добавить(Индекс);
    КонецЦикла;

    ВыдачаКартыПротивнику(2);
    ВыдачаКартыИгроку(2);

    Элементы.НачатьИгру.Доступность = Ложь;
    Элементы.НачатьИгру.ЦветФона = Новый Цвет(195,
195, 195);
    Элементы.НачатьИгру.Заголовок = "Ваш Ход";
    Элементы.ВзятьКарту.Доступность = Истина;
    Элементы.ВзятьКарту.ЦветФона = Новый Цвет(0, 255,
255);
    Элементы.Пас.Доступность = Истина;
    Элементы.Пас.ЦветФона = Новый Цвет(0, 255, 255);

    ОпределениеХода = "Игрок";
```

Разберём этот код последовательно. В начале мы присваиваем переменным «СледСлотПротивника» и «СледСлотПротивника» значение 1 и значение 0 для остальных. Это необходимо для того, что “обнулить” значения переменных до первоначальных значений, а также чтобы обозначить переменной тип данных. К сожалению, язык в 1С не может определять тип данных без явного объявления, поэтому при её использовании может быть вызвана ошибка. После идёт объявления Генератора и трёх массивов и булевые значения для «ПасИгрока» и «ПасПротивника».

Далее идут циклы. Первый цикл перебирает все слоты для игрока и противника. Запомните структуру присваивания. Такой метод называется

интерполяцией, которая в 1С используется в квадратных скобках. При конкатенировании надписи “СлотПротивника” с Индексом, который постоянно меняется (от 1 до 9), получается, что мы обращаемся к элементу формы “СлотПротивника(номер)” и вызываем его параметр Картинка, в которую позже присваиваем другую картинку с декорации “Прозрачность”, и так как в той декорации нет картинки, но и слот получает пустую картинку, то есть - ничего. Цикл нужен для того, чтобы убрать все карты со слотов с прошлой игры. Далее используем цикл для добавления одиннадцати карт в колоду последовательно.

После идут две процедуры, которые пока что пусть прост о будут. Далее мы настраиваем Доступность и Цвет кнопкам, попутно определив заголовок и переменную «ХодИгрока».

Начало игры есть, что уже неплохо. Добавив две процедуры для игрока и противника, что позволят производить выдачу карты:

```
&НаКлиенте
Процедура ВыдачаКартыПротивнику(Итераций)
     Для Индекс = 1 По Итераций Цикл
          ТекущаяКарта = ВыборКарты();

         Если НЕ (СледСлотПротивника = 1) Тогда
              Элементы["СлотПротивника" +
СледСлотПротивника].Картинка = Элементы["Карта" +
ТекущаяКарта].Картинка;
              ГлобальныеОчкиПротивника =
ГлобальныеОчкиПротивника + ТекущаяКарта;
              ОчкиПротивника = "? + " +
Строка(ГлобальныеОчкиПротивника) + " / 21";
         Иначе
              Элементы.СлотПротивника1.Картинка =
Элементы. СкрытаяКарта.Картинка;
```

```
            СкрытаяКартаПротивника = ТекущаяКарта;
        КонецЕсли;

        МассивПротивника.Добавить(ТекущаяКарта);
        СледСлотПротивника = СледСлотПротивника + 1;
    КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ВыдачаКартыИгроку(Итераций)
    Для Индекс = 1 По Итераций Цикл
        ТекущаяКарта = ВыборКарты();
        Элементы["СлотИгрока" +
СледСлотИгрока].Картинка = Элементы["Карта" +
ТекущаяКарта].Картинка;
        ГлобальныеОчкиИгрока = ГлобальныеОчкиИгрока +
ТекущаяКарта;
ОчкиИгрока = Строка(ГлобальныеОчкиИгрока) + " / 21";
        Если (ГлобальныеОчкиИгрока > 21) Тогда
Элементы.ОчкиИгрока.ЦветТекста = Новый Цвет(255, 0, 0);
КонецЕсли;
        СледСлотИгрока = СледСлотИгрока + 1;

        Если НЕ (СледСлотИгрока = 1) Тогда
            МассивИгрока.Добавить(ТекущаяКарта);
        КонецЕсли;
    КонецЦикла;
КонецПроцедуры
```

Обе процедуры способны принимать один аргумент – итерацию. Используется она для того, что с самого начала запустить процедуры пару раз.

Это позволит чуть сократить код: вместо вызова функции два раза просто добавить свойство, которое позволяет выполнять инструкцию процедуры несколько раз.

Для начала мы обращаемся к функции ВыборКарты(), которая пока что не присутствует в коде выше. Получив карту из функции, мы рассматриваем случай: была ли это карта первой, если да, тогда для противника скрываем первую карту, а для игрока вы не добавляем карту в его массив-колоду первую карту.

Если карта уже не первая, то идёт немного другая логика. Добавляем картинку к определённому слоту, высчитываем очки глобально и визуально. Эксклюзивно для игрока меняется цвет его очков в зависимости от того, есть ли у него перебор.

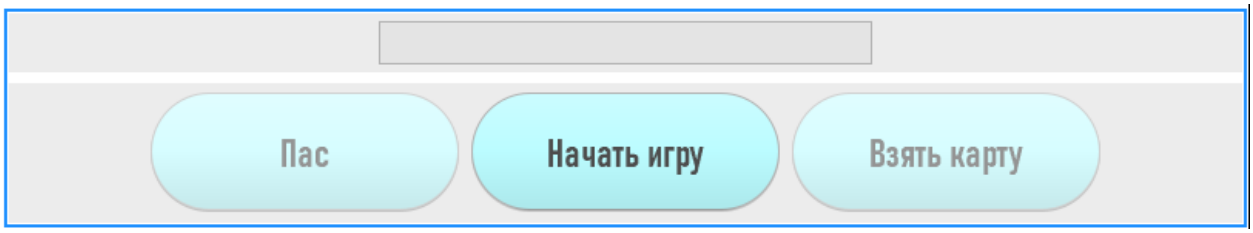
Напишем функцию ВыборКарты():

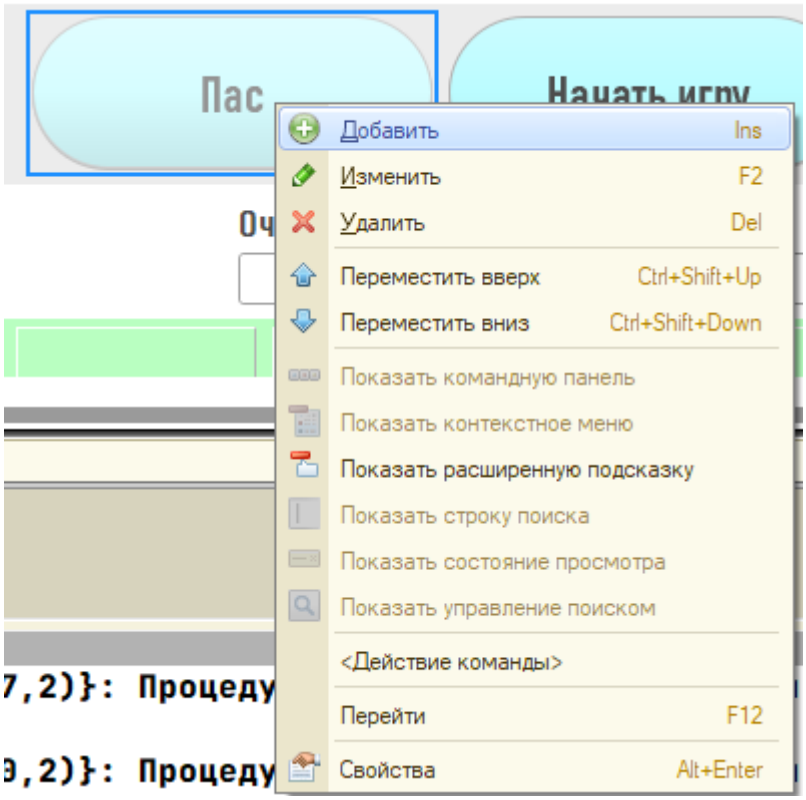
```
&НаКлиенте
Функция ВыборКарты()
    Если Колода.Количество() = 0 Тогда
        Сообщить("Все карты в колоде закончились!");
        Возврат 0;
     КонецЕсли;

    СлучайныйИндекс     =     Генератор.СлучайноеЧисло(0,
Колода.Количество() - 1);
    Карта = Колода[СлучайныйИндекс];
    Колода.Удалить(СлучайныйИндекс);
    Возврат Карта;
КонецФункции
```

В первом случае, если карты в колоде закончили, то карта не будет выдана. Просто вернёт “ничего” и вызовет сообщение. Во втором случае, когда карты ещё есть в колоде, случайным образом выбирается карта, которая

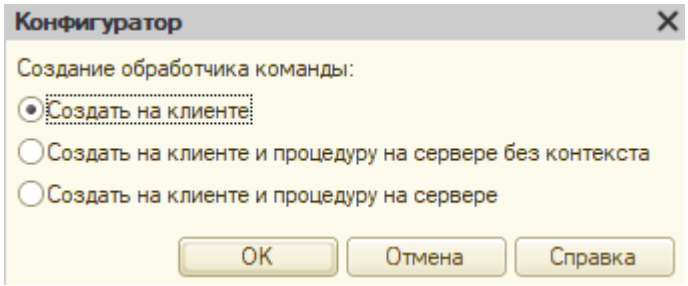
есть в колоде. Записываем в переменную Карта и удаляем её из общей колоды. После возвращаем значение этой карты в одну из двух процедур выше.

Дальше переходим на форму и создаём для кнопок «Пас» и «Взять карту» команду через <Действие команды>

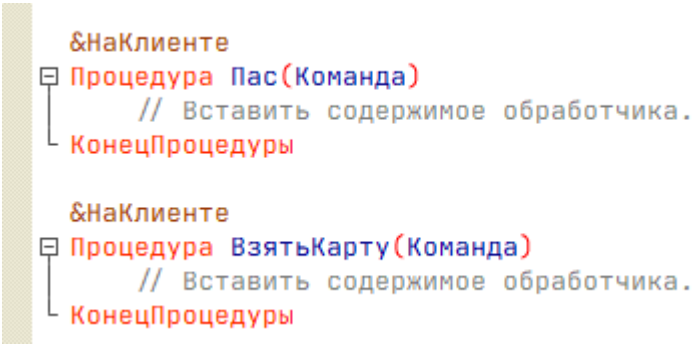




Создаём обработку команды на клиенте:



По итогу должно получится в коде для обоих кнопок:



Вставьте следующий код в процедуру Пас(команда) и процедуру ВзятьКарту(команда):

```
&НаКлиенте
Процедура ВзятьКарту(Команда)
	ПасИгрока = Ложь;
	Элементы. Сообщение.Заголовок = "Вы взяли карту";
	ВыдачаКартыИгроку(1);
	СменаХода();
	КонецИгры();
КонецПроцедуры

&НаКлиенте
Процедура Пас(Команда)
	ПасИгрока = Истина;
	Элементы. Сообщение.Заголовок = "Вы спасовали";
	СменаХода();
	КонецИгры();
КонецПроцедуры
```

Команда взять карту даём игроку карту, меняет его тип действия на Ложь, выдаём сообщение и запускает поочерёдно две процедуры: СменаХода() и КонецИгры().

Создайте процедуру СменаХода() и напиши в ней:

```
&НаКлиенте
Процедура СменаХода()
```

```
    Если (ОпределениеХода = "Игрок") Тогда
        ОпределениеХода = "Противник";
        Элементы.НачатьИгру.Заголовок = "Ход
Противника";
        Элементы.ВзятьКарту.Доступность = Ложь;
        Элементы.Пас.Доступность = Ложь;
        Если (Колода.Количество() = 0) Тогда
            ПодключитьОбработчикОжидания("КонецИгры",
3, Истина);
            Элементы.НачатьИгру.Заголовок   =   "Колода
закончилась!";
        Иначе
            ПодключитьОбработчикОжидания("ИИ",       4,
Истина);
        КонецЕсли;
    ИначеЕсли (ОпределениеХода = "Противник") Тогда
        ОпределениеХода = "Игрок";
        Элементы.НачатьИгру.Заголовок = "Ваш Ход";
        Элементы.ВзятьКарту.Доступность = Истина;
        Элементы.Пас.Доступность = Истина;
    КонецЕсли;
КонецПроцедуры
```

Эта процедура предназначена для того, чтобы менять действующий ход с игрока на противника, и наоборот. Это процедура может отключатьи включать интерфейс выбора игрока через элементы Доступности.

Также обратите внимание на встроенный метод ПодключитьОбработчикОжидания. Этот метод позволяет вызвать процедуру (первый аргумент) через определённое количество секунд (второй аргумент). Таким образом каждое действие происходит не моментально, а с задержкой, что помогает концентрироваться на сообщениях и ведении игры.

Процедура КонецИгры() выглядит следующим образом:

```
&НаКлиенте
Процедура КонецИгры()
    Если (ПасИгрока И ПасПротивника ИЛИ
Колода.Количество() = 0) Тогда
        Элементы.НачатьИгру.Заголовок = "Подсчёт
очков";
        Элементы.ВзятьКарту.Доступность = Ложь;
        Элементы.Пас.Доступность = Ложь;
        ПодключитьОбработчикОжидания("Финал", 4,
Истина);
    КонецЕсли;
КонецПроцедуры
```

В процедуры описывается случай окончания игры, если игрок и противник спасовали или в колоде закончились карты. Также отключает интерфейс и переносит нас в процедуру Финал() с задержкой в 4 секунды.

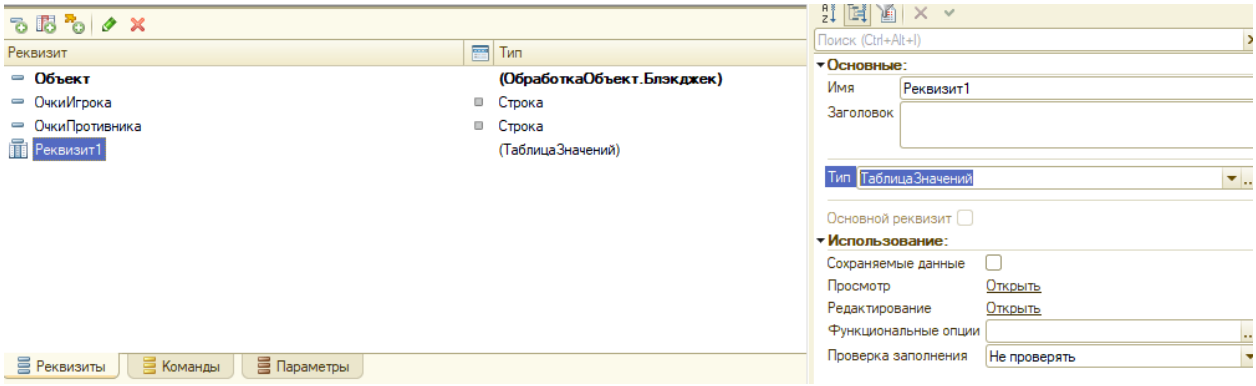
Напишите процедуру Финал():

```
&НаКлиенте
Процедура Финал()
    ГлобальныеОчкиПротивника =
ГлобальныеОчкиПротивника + СкрытаяКартаПротивника;
    ОчкиПротивника = Строка(ГлобальныеОчкиПротивника)
+ " / 21";
    Элементы["СлотПротивника1"].Картинка =
Элементы["Карта" + СкрытаяКартаПротивника].Картинка;

    Если (ГлобальныеОчкиПротивника <
ГлобальныеОчкиИгрока) Тогда
        Если (ГлобальныеОчкиИгрока > 21) Тогда
```

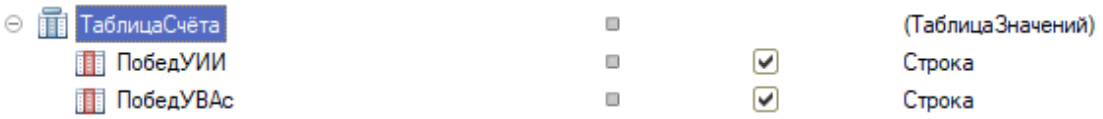
```
            Элементы.НачатьИгру.Заголовок = "ТЫ
ПРОИГРАЛ!";
            ПобедПротивника = ПобедПротивника + 1;
        Иначе
            Элементы.НачатьИгру.Заголовок = "ТЫ
ПОБЕДИЛ!";
            ПобедИгрока = ПобедИгрока + 1;
        КонецЕсли;
    ИначеЕсли (ГлобальныеОчкиПротивника >
ГлобальныеОчкиИгрока) Тогда
        Если (ГлобальныеОчкиПротивника > 21) Тогда
            Элементы.НачатьИгру.Заголовок = "ТЫ
ПОБЕДИЛ!";
            ПобедИгрока = ПобедИгрока + 1;
        Иначе
            Элементы.НачатьИгру.Заголовок = "ТЫ
ПРОИГРАЛ!";
            ПобедПротивника = ПобедПротивника + 1;
        КонецЕсли;
    Иначе
        Элементы.НачатьИгру.Заголовок = "НИЧЬЯ!";
    КонецЕсли;

    НоваяСтрока = ТаблицаСчёта.Добавить();

    НоваяСтрока.ПобедУИИ = ПобедПротивника;
    НоваяСтрока.ПобедУВАс = ПобедИгрока;
    ПодключитьОбработчикОжидания("ИгратьДальше", 3,
Истина);
КонецПроцедуры
```

В процедуре происходит вычисление итоговых очков противника и игрока, а также скрывается первая настоящая карта противника. Здесь же ставится и условия для победы, поражения и ничьи, которые были описаны на первой странице. Для того чтобы функция работа без ошибок добавим табличный вывод побед игрока и противника.

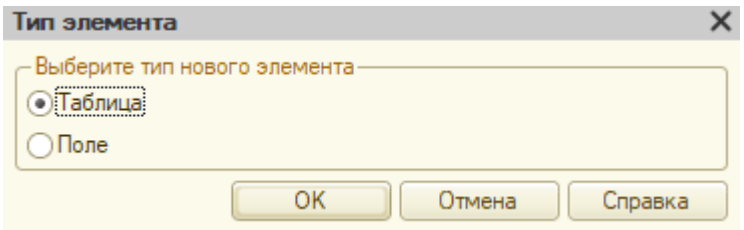
Создаём реквизит на форме с типом данных ТаблицаЗначений:

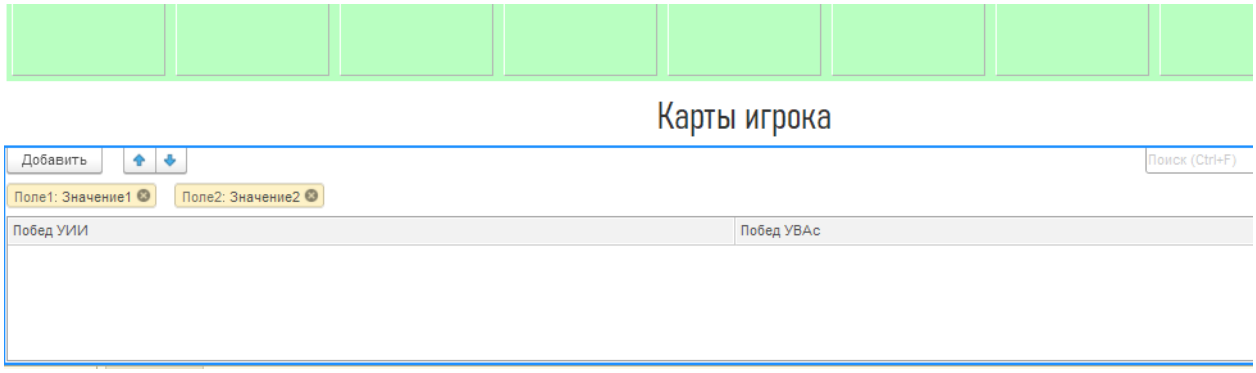


Переименовываем «Реквизит1» в «ТаблицаСчёта» (и да, в этот раз пишем через букву “ё”). Кликаем по реквизиты ПКМ и выбираем два раза добавить столбец. Переименовываем столбцы следующим образом:



Переносим таблицу на форму (в левую часть). Выбираем тип Таблица и создаём колонки. Переносим таблицу в самый вниз формы





Самостоятельно создайте процедуру Играть Дальше() и впишите в нём инструкцию. Вам необходимо менять доступность главной кнопки, сбрасывать цвет очков или менять иные необходимые параметры.

С блоком базовой логики разобрались. Далее необходимо создать инструкцию и логику ИИ.

6. Реализация логики ИИ противника

ИИ – это технология, что мыслит алгоритмами и процессами, и пытается в некотором роде подражать человеку, принимая решения и делая определённые выводы исходя из входных данных, тем самым высчитывая закономерности.

Наш ИИ будет примитивным и не сложным. Чтобы реализовать его нам потребуется провести три необходимых этапа:

- приём входных данных;
- манипулирование входными данными;
- принятие решения.

Создадим процедуру ИИ() (обязательно поместите её между процедурой СменаХода() и КонецИгры()), распишем первый этап кода:

```
    Если (ПасИгрока И ПасПротивника) Тогда Return;
КонецЕсли;
        Решение = 0;
         // Сбор Данных
         Для Каждого Элемент Из МассивПротивника Цикл
             АбстрактныйСчётПротивника                    =
АбстрактныйСчётПротивника + Элемент;
         КонецЦикла;

        АбстрактныйСчётИгрока = 0;
         Для Каждого Элемент Из МассивИгрока Цикл
             АбстрактныйСчётИгрока                        =
АбстрактныйСчётИгрока + Элемент;
```

```
        КонецЦикла;

        АбстрактныйОбщийСчётЗначенийВКолоде = Новый
Массив;
        ВсеКарты = Новый Массив;
        Для Индекс = 1 По 11 Цикл
            ВсеКарты.Добавить(Индекс);
        КонецЦикла;

        Для Каждого Элемент Из МассивПротивника Цикл
            НайденныйИндекс = ВсеКарты.Найти(Элемент);
            Если НайденныйИндекс <> Неопределено Тогда
                ВсеКарты.Удалить(НайденныйИндекс);
            КонецЕсли;
        КонецЦикла;

        Для Каждого Элемент Из МассивИгрока Цикл
            НайденныйИндекс = ВсеКарты.Найти(Элемент);
            Если НайденныйИндекс <> Неопределено Тогда
                ВсеКарты.Удалить(НайденныйИндекс);
            КонецЕсли;
        КонецЦикла;

        АбстрактныйОбщийСчётЗначенийВКолоде = ВсеКарты;
```

В начале ставим условие, что, если оба игрока уже спасовали, то ИИ принимать решение не нужно. После создаём локальную переменную «Решение», которая принимает в себя два значения 0 или 1, соответственно Пас или Взять карту. В этом коде противник вычисляет примерные карты в колоде, где мы специально исключаем первую карту игрока. Также противник анализирует свои карты и свои текущие очки.

Перейдём ко второму этапу и напишем код, где ИИ будет манипулировать входными данными:

```
     // Сбор Тактических данных
       Если (АбстрактныйСчётПротивника >
АбстрактныйСчётИгрока) Тогда
        АнализСчёта = -0.1;
        РазницаОчков = АбстрактныйСчётПротивника -
АбстрактныйСчётИгрока;
       ИначеЕсли (АбстрактныйСчётПротивника =
АбстрактныйСчётИгрока) Тогда
        АнализСчёта = 0;
        РазницаОчков = 0;
       ИначеЕсли (АбстрактныйСчётПротивника <
АбстрактныйСчётИгрока) Тогда
        АнализСчёта = 0.1;
        РазницаОчков = АбстрактныйСчётИгрока -
АбстрактныйСчётПротивника;
       КонецЕсли;

       ДоМаксимумаПротивнику = 21 -
АбстрактныйСчётПротивника;
       ДоМаксимумаИгроку = 21 - АбстрактныйСчётИгрока;

       Для Каждого Элемент Из
АбстрактныйОбщийСчётЗначенийВКолоде Цикл
        Если (АбстрактныйСчётПротивника + Элемент >=
21) Тогда АнализУспехаПротивника =
АнализУспехаПротивника + 1;
        КонецЕсли;
       КонецЦикла;
```

```
        УспехПротивника = АнализУспехаПротивника /
АбстрактныйОбщийСчётЗначенийВКолоде.Количество();

        Для Каждого Элемент Из
АбстрактныйОбщийСчётЗначенийВКолоде Цикл
         Если (АбстрактныйСчётИгрока + Элемент >= 21)
Тогда АнализУспехаИгрока = АнализУспехаИгрока + 1;
         КонецЕсли;
        КонецЦикла;

        УспехИгрока = АнализУспехаИгрока /
АбстрактныйОбщийСчётЗначенийВКолоде.Количество();

        Если ПасИгрока Тогда
              ДопПрибавка = 0.2;
         МинимальныйПорог = 0.3;
        Иначе
              ДопПрибавка = 0;
              МинимальныйПорог = 0.4;
        КонецЕсли;

        ВесУспеха = 0.95 + ДопПрибавка;
        ВесСчета = 0.85 + ДопПрибавка;
        ВесБлизостиКПобеде = 0.25 + ДопПрибавка;
```

Мы могли взять абсолютно любые возможные случае расчёта. В коде представлены следующие основные категории: кто одерживает вверх, какая разница очков между игроком и противником, какой успех у противника взять оптимальную карту, а какой у игрока, сколько максимум очков необходимо игроку или противнику до двадцати одного, спасовал ли игрок перед ходом противника. Дополнительно к этому имеются веса параметров (их

приоритетность) и минимальный порог (если значение выше минимального порога значит можно взять карту).

Последние: произвести финальный расчёт и обозначить действия:

```
// Расчет коэффициента
     ОбщийКоэффициент = (УспехПротивника * УспехИгрока
* ВесУспеха) + (АнализСчёта * РазницаОчков * ВесСчета)
+ ((ДоМаксимумаПротивнику / (ДоМаксимумаИгроку + 0.01)
* 0.25) * ВесБлизостиКПобеде);

     Если АбстрактныйСчётИгрока >= 21 ИЛИ
АбстрактныйСчётПротивника >= 21 Тогда
     Решение = 0;
     ИначеЕсли АбстрактныйСчётПротивника <
АбстрактныйСчётИгрока Тогда
        Если ОбщийКоэффициент > МинимальныйПорог Тогда
         Решение = 1;
     Иначе
         Решение = 0;
     КонецЕсли;
     ИначеЕсли АбстрактныйСчётПротивника >
АбстрактныйСчётИгрока Тогда
         Если ОбщийКоэффициент > 0.6 Тогда
             Решение = 1;
         Иначе
         Решение = 0;
     КонецЕсли;
     Иначе
     Если ОбщийКоэффициент > 0.5 Тогда
         Решение = 1;
         Иначе
```

```
            Решение = 0;
        КонецЕсли;
        КонецЕсли;

        Если (Решение = 1) Тогда
            ПасПротивника = Ложь;
        Элементы. Сообщение.Заголовок = "ИИ взял карту";
            ВыдачаКартыПротивнику(1);
ПодключитьОбработчикОжидания("СменаХода", 2, Истина);
            КонецИгры()
        Иначе
            ПасПротивника = Истина;
            Элементы. Сообщение.Заголовок = "ИИ спасовал";
            Если НЕ (ПасПротивника И ПасИгрока) Тогда
ПодключитьОбработчикОжидания("СменаХода", 2, Истина);
            КонецЕсли;
            КонецИгры()
        КонецЕсли;
```

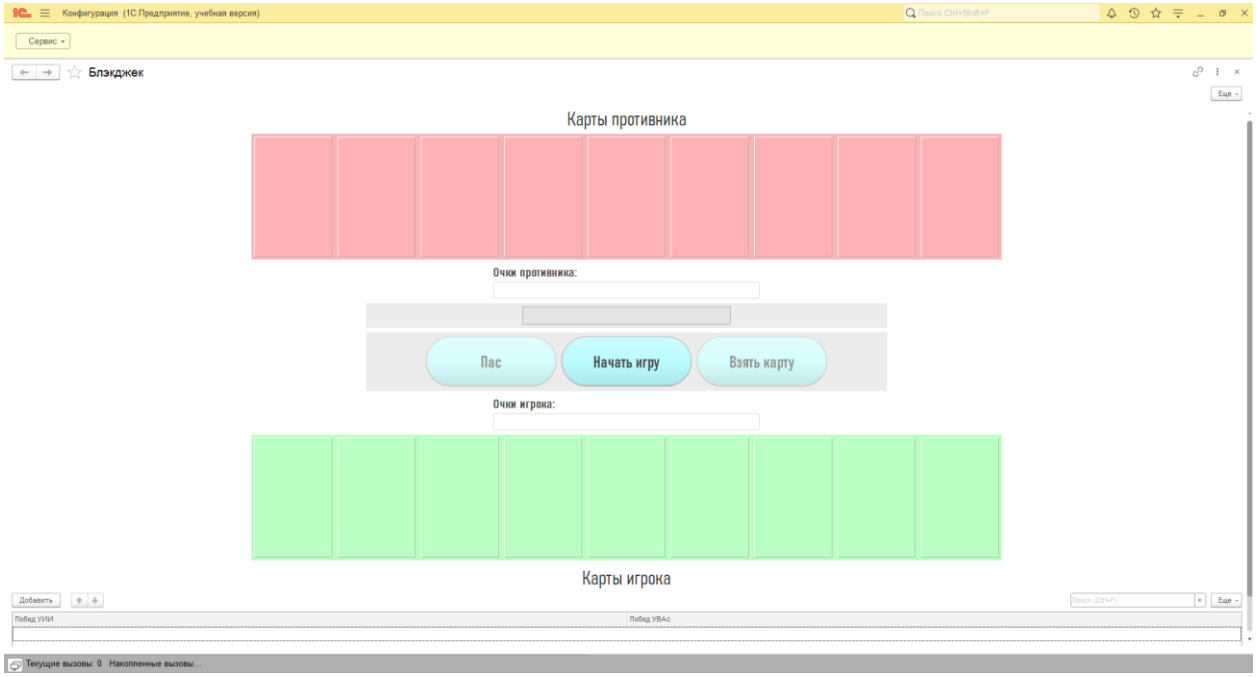
Общий коэффициент – это универсальная формула подсчёта всех параметров, которая выводит, в основном, значения от 0 до 1. Далее пишутся все определённые действия, которые может выполнить ИИ. Всего их два, однако они могут выполняться в различных случаях. Есть те, где вообще не требуется коэффициент формулы, к примеру, если у противника уже 21 очко.

Вот такая вот примитивная и небольшая логика ИИ. Если вы хотите, то можете видоизменить формулу или вывести свою – более оптимальную и качественную.

7. **Тестирование**

Запустите конфигурацию и протестируйте. При наличии ошибок проверьте код, форму и названия переменных и реквизитов. Постарайтесь

исправить ошибки самостоятельно. Чтобы изменить сложность ИИ (сделать так, чтобы он пасовал не так часто) можно повысить на 0.05 или 0.1 вес успеха или счёта или близости к успеху, к примеру, ВесУспеха с 0.95 до 1.



Таким образом вы успешно воссоздали «21 очко» в обработчике 1С. Теперь вы можете произвести улучшения игры или просто сдать работу.

Разработка игры «Виселица»
Игра разработана совместно со студентом
группы ИС-31 Вдовиным Никитой

Цель: разработка полнофункциональной игры «Виселица» в конфигураторе «1С:Предприятие 8» с возможностью выбора режима игры (против компьютера или против другого игрока), визуальным отображением прогресса (картинка виселицы), ведением статистики игроков и интуитивным пользовательским интерфейсом.

Вопросы для обсуждения перед разработкой

1. Правила игры в Виселицу. Один из игроков загадывает слово — пишет на бумаге любые две буквы слова и отмечает места для остальных букв. Второй игрок предлагает букву, которая может входить в это слово. Если такая буква есть в слове, то первый игрок пишет её над соответствующими этой букве чертами — столько раз, сколько она встречается в слове. Если такой буквы нет, то к виселице добавляется круг в петле, изображающий голову. Второй игрок продолжает отгадывать буквы до тех пор, пока не отгадает всё слово. За каждый неправильный ответ первый игрок добавляет одну часть туловища к виселице. Если туловище в виселице нарисовано полностью, то отгадывающий игрок проигрывает. Если игроку удаётся угадать слово, он выигрывает и может загадывать слово.
2. Как должно меняться изображение, сколько необходимо картинок.
3. Откуда появляется загаданное слово. Где хранится набор слов.
4. Когда «загадывается» слово, как показать игроку, сколько букв в этом слове.
5. Как игроку вводить буквы (поле ввода, кнопки), как организовать исключение уже названных букв, чтобы игрок не мог второй раз выбрать одну и ту же букву.

6. Можно ли играть игроку с игроком, когда один загадывает слово, другой разгадывает.
7. Возможно ли организовывать игру с «компьютером», когда игрок загадывает слово, а «компьютер» разгадывает.

По результатам обсуждения можно сформулировать следующие задачи:

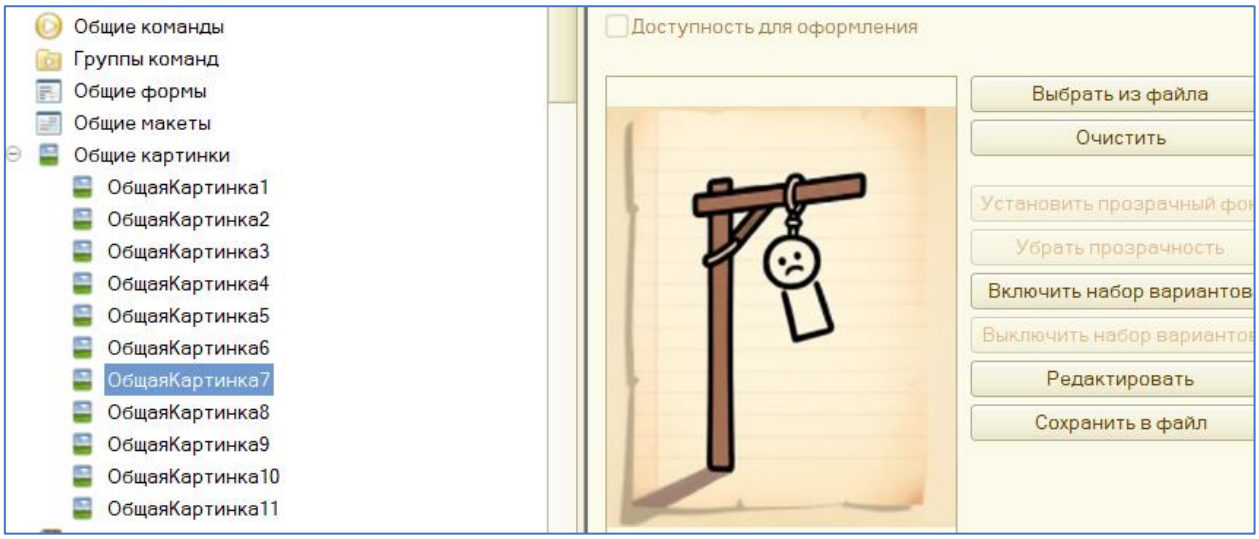
Задачи

1. Реализовать два режима игры:
    - **С компьютером** — система автоматически загадывает слово из справочника.
    - **С игроком** — один игрок вводит слово, второй угадывает.
2. Обеспечить корректную обработку ввода, валидацию данных и защиту от ошибок.
3. Визуализировать процесс игры:
    - Отображение текущего состояния слова (через подчеркивания и открытые буквы).
    - Постепенное построение изображения виселицы при ошибках.
4. Вести статистику побед и поражений для каждого игрока (только в режиме с компьютером).
5. Реализовать удобный интерфейс с кнопками букв, панелью управления и возможностью сброса игры.

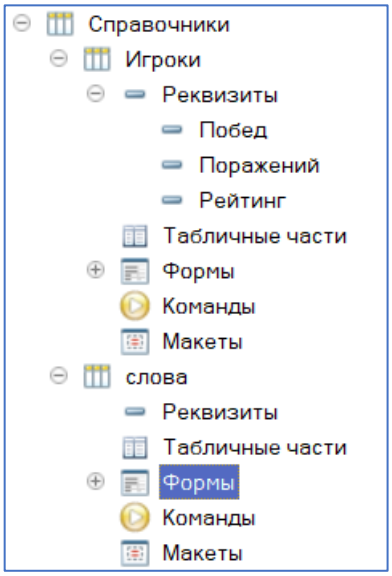
Ход разработки

1. Разработка интерфейса игры

Добавьте изображения в Общие картинки (количество вариантов изображений 11).

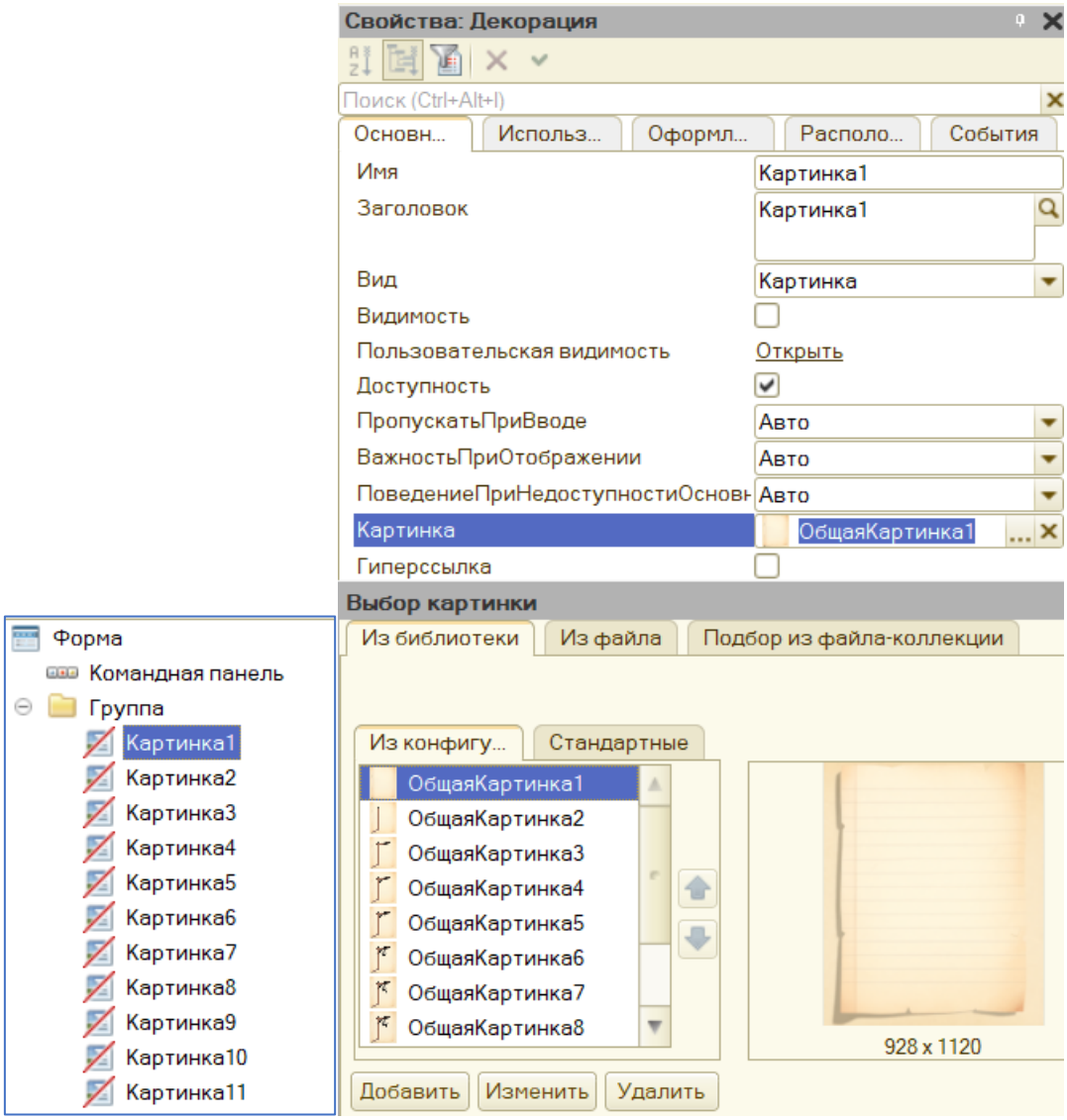


Создайте справочники для хранения информации об игроках и словах.

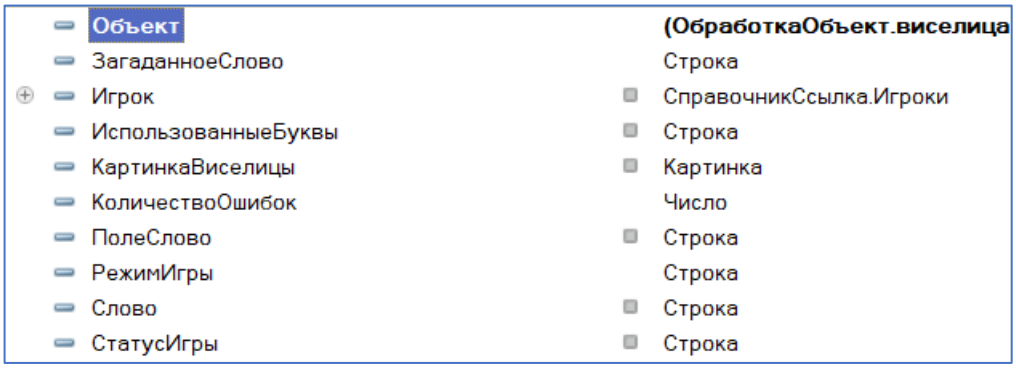


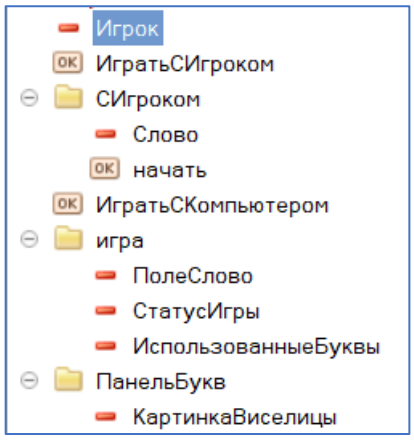
Создайте обработку, форму обработки, основные элементы формы

Создайте группу, в которую добавьте Декорации (картинки). Настройте соответствие Декорации и общей картинки.



Создайте объекты (реквизиты) и перенесите на форму.





Поле ввода "Слово"

- Имя: Слово
- Тип: ПолеВвода
- Подпись: Слово: (или просто Слово)
- Доступность: Ложь (по умолчанию — активируется только в режиме «с игроком»)

Поле ввода "Использованные буквы"

- Имя: ИспользованныеБуквы
- Тип: ПолеВвода
- Подпись: Использованные буквы
- Доступность: Ложь

Поле "Статус игры"

- Имя: СтатусИгры
- Тип: ПолеЗначения
- Подпись: Статус игры:
- Значение по умолчанию: Выберите игрока и режим игры

Поле "Поле слово"

- Имя: ПолеСлово
- Тип: ПолеЗначения
- Подпись: Поле слово:
- Значение по умолчанию: пустая строка

  Создайте команды формы и перенесите их на область формы.

Кнопка "Играть с компьютером"

- Имя: ИгратьСКомпьютером
- Тип: Кнопка
- Надпись: Играть с компьютером
- Доступность: Ложь (активируется после выбора игрока)

Кнопка "Играть с игроком"

- Имя: ИгратьСИгроком
- Тип: Кнопка
- Надпись: Играть с игроком
- Доступность: Ложь

Кнопка "Начать"

- Имя: Начать
- Тип: Кнопка
- Надпись: Начать
- Доступность: Ложь

Кнопка "Показать рейтинг"

- Имя: КнопкаПоказатьРейтинг
- Тип: Кнопка
- Надпись: Показать рейтинг

Создайте группу для букв — ПанельБукв. Это будет основной контейнер для всех 33 кнопок букв, разбитых на 5 строк.

- Имя: ПанельБукв
- Тип: Группа
- Расположение: ниже поля ПолеСлово
- Свойства:
  - **ГоризонтальноеПоложение**: Лево
  - **ВертикальноеПоложение**: Верх
  - **Ширина**: Автоматически
  - **Высота**: Автоматически

Внутри этой группы нужно создать 6 вложенных групп — по одной на строку (или столбец).

Назовите их:

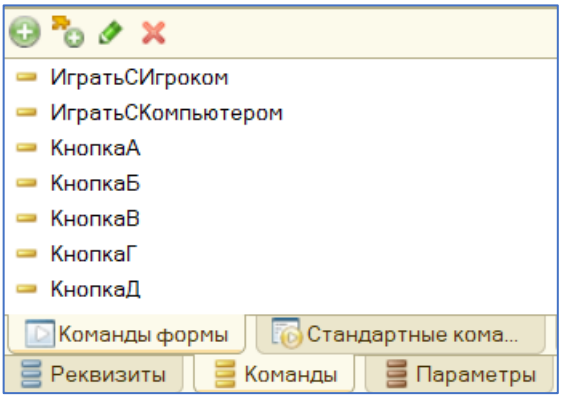
- ГруппаБукв1
- ГруппаБукв2
- ГруппаБукв3
- ГруппаБукв4
- ГруппаБукв5
- ГруппаБукв6

Для каждой группы установите:

- **ГоризонтальноеПоложение**: Лево
- **ВертикальноеПоложение**: Верх
- **Ширина**: Автоматически
- **Высота**: Автоматически

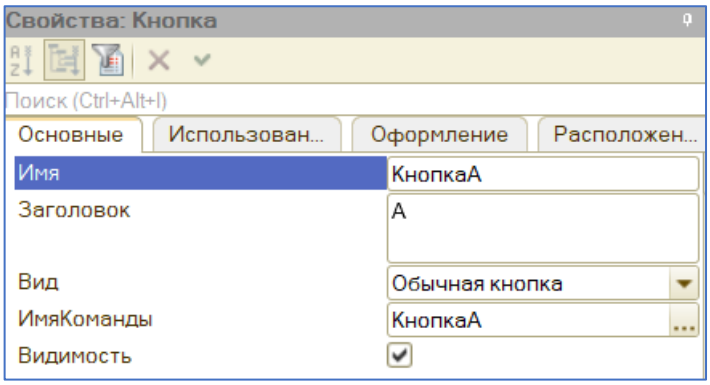
Эти группы будут содержать кнопки букв и автоматически располагаться одна под другой.

Добавьте Команды формы в виде букв.



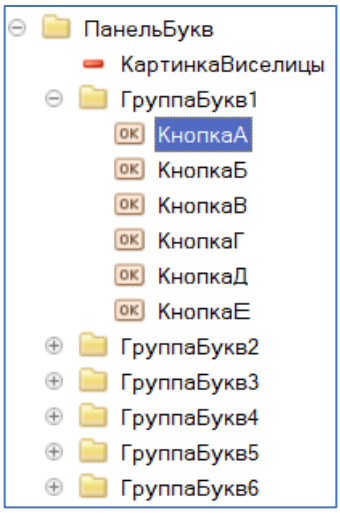
Создайте кнопки букв внутри каждой группы. В каждой из 6 групп создайте кнопки по алфавиту.

Настройте свойства кнопок. Для **каждой кнопки** (например, КнопкаА) установите:



- Имя: КнопкаА
- ГоризонтальноеПоложение: Лево
- ВертикальноеПоложение: Центр

Чтобы кнопки не прилипали друг к другу — добавьте небольшие отступы.



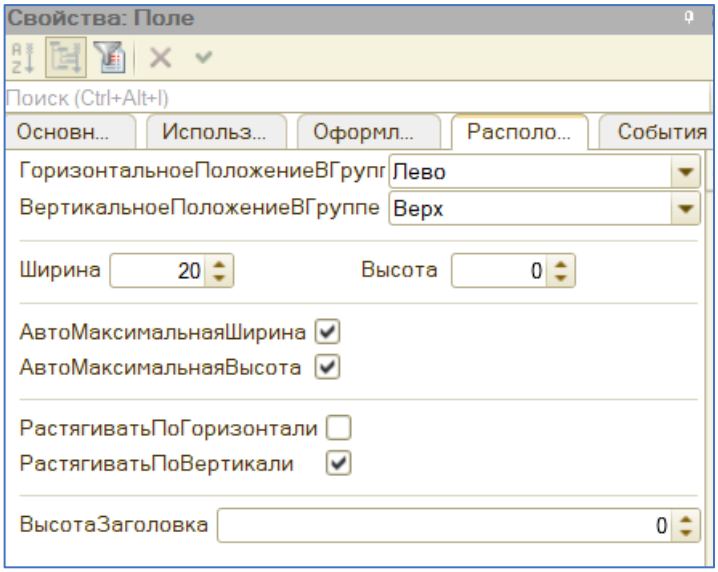
Добавьте картинку виселицы (в примере картинку на одном уровне с буквами, поэтому в той же группе_

Создайте реквизит формы:

- Имя: КартинкаВиселицы
- Тип: Картинка
- Расположение: справа от поля ПолеСлово или под ним
- Свойства:
  - **Размер картинки**: Пропорционально

В дальнейшем коде используется КартинкаВиселицы = ПолучитьКартинкуВиселицы(КоличествоОшибок); — значит, картинка

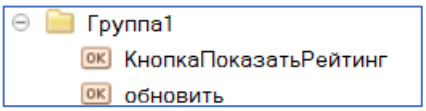
должна быть **одним элементом**, который меняет своё значение (картинку) в зависимости от ошибок.



Создайте поле выбора игрока

- Имя: Игрок
- Тип: ПолеВыбора
- Подпись: Игрок:
- Ссылка на справочник: Справочник.Игроки
- Доступность: Истина (по умолчанию)

Ориентируясь на изображение интерфейса добавьте недостающие элементы. Помните, что вы можете разработать свой интерфейс или расположить элементы, как вам нравится больше.



Игрок:

Играть с игроком

Слово: Начать

Играть с компьютером

Игра

Поле слово:

Статус игры:

Использованные буквы:

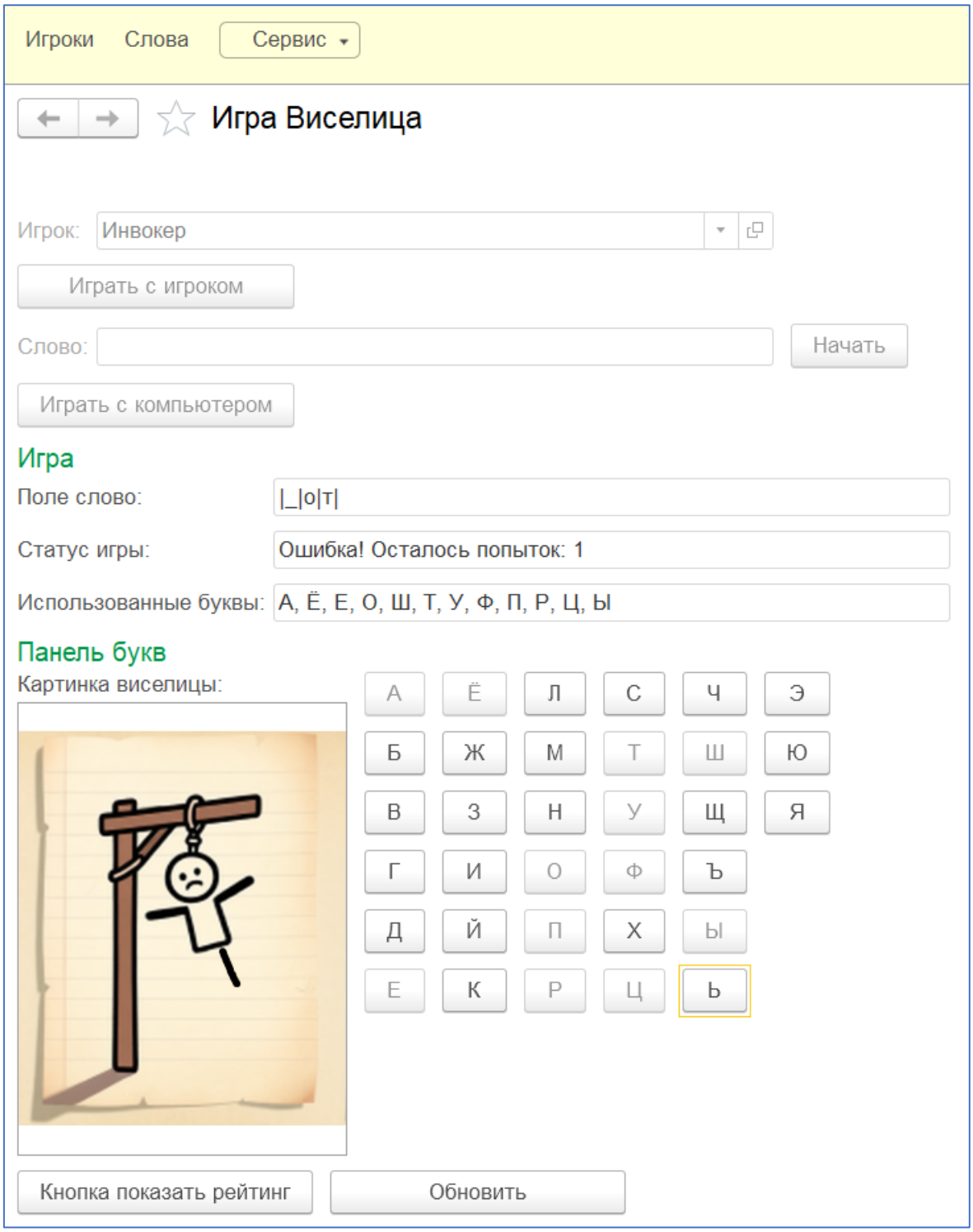
Панель букв

Картинка виселицы:

А	Ё	Л	С	Ч	Э
Б	Ж	М	Т	Ш	Ю
В	З	Н	У	Щ	Я
Г	И	О	Ф	Ъ	
Д	Й	П	Х	Ы	
Е	К	Р	Ц	Ь	

Кнопка показать рейтинг

Обновить



Программирование игры

Модуль формы обработки игры «Виселица»

1. Процедура ПриОткрытии(Отказ)

Назначение

Инициализирует начальное состояние игры при открытии формы: сбрасывает все реквизиты и обновляет доступность элементов управления.

Алгоритм

1. Очищает все реквизиты (слово, ошибки, статус и т.д.).

2. Вызывает ОбновитьДоступность() для правильного отображения интерфейса.

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
Инициализируем реквизиты — возвращаем форму в "чистое" состояние
ЗагаданноеСлово = "";// Слово, которое нужно угадать
КоличествоОшибок = 0;// Счётчик ошибок (максимум — 11)
РежимИгры = "";// Пустая строка означает, что режим ещё не выбран
Слово = "";// Поле ввода слова (для режима "Игрок")
ПолеСлово = "";// Отображаемое поле с подчёркиваниями и открытыми буквами
СтатусИгры = "";// Текстовое сообщение пользователю
ИспользованныеБуквы = "";// Строка вида "А, Б, В" — для истории
КартинкаВиселицы = Неопределено; // Нет изображения до первой ошибки
ОбновитьДоступность();   //   Настроить   видимость   и доступность кнопок и полей
КонецПроцедуры
```

2. Процедура ИгрокПриИзменении(Элемент)

Назначение

Реагирует на выбор игрока в поле `Игрок` и обновляет доступность кнопок выбора режима игры.

Алгоритм

1. При изменении значения поля «Игрок» вызывается `ОбновитьДоступность()`.

2. Если игрок выбран — кнопки режимов становятся доступны.

```
&НаКлиенте
Процедура ИгрокПриИзменении(Элемент)
ОбновитьДоступность(); // Обновить интерфейс при выборе игрока
КонецПроцедуры
```

3. Процедура ОбновитьДоступность()

Назначение

Управляет видимостью и доступностью элементов управления в зависимости от текущего состояния игры.

Алгоритм

1. Проверяет, выбран ли игрок.

2. Если да — делает доступными кнопки выбора режима.

3. По умолчанию скрывает игровые элементы (`Игра`, `ПанельБукв`, `Слово`, `Начать`).

```
&НаКлиенте
Процедура ОбновитьДоступность()
ИгрокВыбран = Игрок <> Неопределено; // Игрок выбран?
// Кнопки режимов доступны, только если игрок выбран
Элементы.ИгратьСИгроком.Доступность = ИгрокВыбран;
Элементы.ИгратьСКомпьютером.Доступность = ИгрокВыбран;
// Игровые элементы недоступны до выбора режима
Элементы.Игра.Доступность = Ложь;
Элементы.ПанельБукв.Доступность = Ложь;
Элементы.Слово.Доступность = Ложь;
Элементы.Начать.Доступность = Ложь;
```

```
КонецПроцедуры
```

4. Процедура `ИгратьСИгроком(Команда)`

Назначение

Активирует режим игры «с другим игроком»: один вводит слово, второй угадывает.

Алгоритм

1. Устанавливает статус и режим игры.
2. Блокирует выбор игрока и режимов.
3. Делает доступными поле ввода слова и кнопку «Начать».

```
&НаКлиенте
Процедура ИгратьСИгроком(Команда)
    СтатусИгры = "Режим: Игра с игроком";
    РежимИгры = "Игрок";
    // Блокируем выбор режима и игрока
    Элементы.ИгратьСИгроком.Доступность = Ложь;
    Элементы.ИгратьСКомпьютером.Доступность = Ложь;
    Элементы.Игрок.Доступность = Ложь;
    // Разрешаем ввод слова и запуск игры
    Элементы.Слово.Доступность = Истина;
    Элементы.Начать.Доступность = Истина;
КонецПроцедуры
```

5. Процедура `ИгратьСКомпьютером(Команда)`

Назначение

Активирует режим игры против компьютера: слово выбирается случайно из справочника.

Алгоритм

1. Устанавливает статус и режим.
2. Блокирует выбор игрока и режимов.

3. Активирует игровую область и панель букв.

4. Инициализирует игру с помощью серверной функции.

```
&НаКлиенте
Процедура ИгратьСКомпьютером(Команда)
    СтатусИгры = "Режим: Игра с компьютером";
    РежимИгры = "Компьютер";
    // Блокировка выбора
    Элементы.ИгратьСИгроком.Доступность = Ложь;
    Элементы.ИгратьСКомпьютером.Доступность = Ложь;
    Элементы.Игрок.Доступность = Ложь;
    // Открываем игровую зону
    Элементы.Игра.Доступность = Истина;
    Элементы.ПанельБукв.Доступность = Истина;
    ИнициализироватьИгруСКомпьютером();   //   Получить
слово и начать
КонецПроцедуры
```

6. Процедура Начать(Команда)

Назначение

Запускает игру в режиме «с игроком» после ввода слова.

Алгоритм

1. Проверяет, введено ли слово.

2. Проверяет, что слово состоит только из букв.

3. Если всё корректно — инициализирует игру, скрывает поле ввода.

```
&НаКлиенте
Процедура Начать(Команда)
    Если ПустаяСтрока(Слово) Тогда
        СтатусИгры = "Введите слово для игры!";
        Возврат;
    КонецЕсли;
```

```
    Если Не ПроверитьСлово(Слово) Тогда
    СтатусИгры = "Слово должно содержать только буквы!";
        Возврат;
    КонецЕсли;
    // Скрываем ввод
    Элементы.Слово.Доступность = Ложь;
    Элементы.Начать.Доступность = Ложь;
    // Открываем игру
    Элементы.Игра.Доступность = Истина;
    Элементы.ПанельБукв.Доступность = Истина;
    ИнициализироватьИгруСИгроком(Слово);
    Слово = ""; // Очищаем поле ввода
КонецПроцедуры
```

7. Процедура `КнопкаПоказатьРейтинг(Команда)`

Назначение

Открывает форму справочника «Игроки» для просмотра статистики.

Алгоритм

Просто вызывает стандартную форму списка справочника.

```
&НаКлиенте
Процедура КнопкаПоказатьРейтинг(Команда)
    ОткрытьФорму("Справочник.Игроки.ФормаСписка");
КонецПроцедуры
```

8. Процедура Обновить(Команда) и СброситьИгру()

Назначение

Полный сброс игры к начальному состоянию (аналог «Новая игра»).

Алгоритм

1. Очищает все реквизиты.

2. Разблокирует все буквы.

3. Восстанавливает доступ к выбору игрока и режимов.

```
&НаКлиенте
Процедура Обновить(Команда)
    СброситьИгру();
КонецПроцедуры
&НаКлиенте
Процедура СброситьИгру()
    ЗагаданноеСлово = "";
    КоличествоОшибок = 0;
    РежимИгры = "";
    Слово = "";
    ПолеСлово = "";
    СтатусИгры = "Выберите игрока и режим игры";
    ИспользованныеБуквы = "";
    КартинкаВиселицы = Неопределено;
    РазблокироватьВсеБуквы(); // Все буквы снова активны
    ОбновитьДоступность();      // Вернуть интерфейс к
начальному виду
    Элементы.Игрок.Доступность   =   Истина;   //   Можно
выбрать игрока снова
    ИгрокВыбран = Игрок <> Неопределено;
    Элементы.ИгратьСИгроком.Доступность = ИгрокВыбран;
    Элементы.ИгратьСКомпьютером.Доступность            =
ИгрокВыбран;
КонецПроцедуры
```

9. Процедуры работы с буквами: РазблокироватьВсеБуквы(), ЗаблокироватьВсеБуквы(), ПолучитьБукву ПоИндексу()

Назначение

Управление доступностью кнопок букв в зависимости от состояния игры.

Алгоритм

1. Цикл по 33 буквам русского алфавита.

2. Имя кнопки формируется как `"Кнопка" + Буква`.

3. Используется `Попытка...Исключение` на случай отсутствия кнопки (например, "Ё").

```
&НаКлиенте
Процедура РазблокироватьВсеБуквы()
    Для i = 1 По 33 Цикл
        Буква = ПолучитьБукву ПоИндексу(i);
        Попытка
        Элементы["Кнопка" + Буква].Доступность = Истина;
        Исключение // Кнопка может отсутствовать
(например, Ё)
        КонецПопытки;
    КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ЗаблокироватьВсеБуквы()
    Для i = 1 По 33 Цикл
        Буква = ПолучитьБукву ПоИндексу(i);
        Попытка
            Элементы["Кнопка" + Буква].Доступность =
Ложь;
        Исключение
        КонецПопытки;
    КонецЦикла;
КонецПроцедуры
```

```
&НаКлиенте
Функция ПолучитьБукву ПоИндексу(Индекс)
    Алфавит =
"А,Б,В,Г,Д,Е,Ё,Ж,З,И,Й,К,Л,М,Н,О,П,Р,С,Т,У,Ф,Х,Ц,Ч,Ш,Щ,
Ъ,Ы,Ь,Э,Ю,Я";
    МассивБукв = СтрРазделить(Алфавит, ",");
    Если Индекс >= 1 И Индекс <= МассивБукв.Количество()
Тогда
        Возврат МассивБукв[Индекс - 1]; // Массив с 0, а
индекс с 1
    Иначе
        Возврат "";
    КонецЕсли;
КонецФункции
```

10. Функция ПроверитьСлово(ПроверяемоеСлово)

Проверяет, состоит ли слово только из букв (русских и латинских, включая «Ё»).

```
&НаКлиенте
Функция ПроверитьСлово(ПроверяемоеСлово)
    Для i = 1 По СтрДлина(ПроверяемоеСлово) Цикл
        Символ = Сред(ПроверяемоеСлово, i, 1);
        // Разрешены только буквы
        Если
Найти("ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФ
ХЦЧШЩЪЫЬЭЮЯ", ВРег(Символ)) = 0 Тогда
            Возврат Ложь;
        КонецЕсли;
    КонецЦикла;
    Возврат Истина;
```

```
КонецФункции
```

11. Процедуры инициализации игры: ИнициализироватьИгруСИгроком(),ИнициализироватьИгруСКомпьютером()

Назначение

Подготавливают начальное состояние игры: маска слова, статус, сброс ошибок.

```
&НаКлиенте
Процедура ИнициализироватьИгруСИгроком(ВведенноеСлово)
    ЗагаданноеСлово = ВведенноеСлово;
    КоличествоОшибок = 0;
    ТекущееСостояние = "";
    Для i = 1 По СтрДлина(ЗагаданноеСлово) Цикл
        ТекущееСостояние = ТекущееСостояние + "|_";
    КонецЦикла;
    ТекущееСостояние = ТекущееСостояние + "|"; // Закрывающая черта
    ПолеСлово = ТекущееСостояние;
    СтатусИгры = "Угадайте слово! Количество букв: " + СтрДлина(ЗагаданноеСлово);
    ИспользованныеБуквы = "";
    КартинкаВиселицы = Неопределено;
КонецПроцедуры
```

```
&НаКлиенте
Процедура ИнициализироватьИгруСКомпьютером()
    ЗагаданноеСлово = ПолучитьСлучайноеСлово();
    КоличествоОшибок = 0;
    Если ЗагаданноеСлово <> "" Тогда
```

```
        // Аналогично режиму с игроком
        ТекущееСостояние = "";
        Для i = 1 По СтрДлина(ЗагаданноеСлово) Цикл
            ТекущееСостояние = ТекущееСостояние + "|_";
        КонецЦикла;
        ТекущееСостояние = ТекущееСостояние + "|";
        ПолеСлово = ТекущееСостояние;
        СтатусИгры = "Угадайте слово! Количество букв: "
+ СтрДлина(ЗагаданноеСлово);
        ИспользованныеБуквы = "";
        КартинкаВиселицы = Неопределено;
    Иначе
        СтатусИгры = "Ошибка: в справочнике нет слов!";
    КонецЕсли;
КонецПроцедуры
```

12. Процедура КнопкаБуквыНажатие(Элемент)

Назначение

Универсальный обработчик нажатия любой буквенной кнопки (формата КнопкаА, КнопкаБ и т.д.).

Алгоритм

1. Получить имя нажатой кнопки из свойства Элемент.Имя.

2. Извлечь букву, начиная с 7-го символа строки (т.к. префикс "Кнопка" состоит из 6 символов).

3. Передать извлечённую букву в процедуру логики игры ОбработатьНажатуюБукву.

4. Отключить нажатую кнопку, чтобы её нельзя было нажать повторно.

```
&НаКлиенте
Процедура КнопкаБуквыНажатие(Элемент)
    ИмяКнопки = Элемент.Имя;
```

```
    Буква = Сред(ИмяКнопки, 7); // "КнопкаА" → "А"
    ОбработатьНажатуюБукву(Буква);
    Элементы[ИмяКнопки].Доступность = Ложь; // Буква
больше не выбирается
КонецПроцедуры
```

13. Процедура ОбработатьНажатуюБукву(Буква)

Назначение

Реализует основную логику проверки буквы и реакции на результат.

Алгоритм

1. Добавить букву в строку использованных букв (через запятую).

2. Проверить, содержится ли буква (без учёта регистра) в загаданном слове.

Если ДА:

Вызвать процедуру открытия буквы в поле слова.

Проверить, остались ли в поле слова подчёркивания (_).

Если НЕТ → игрок угадал слово:

Установить статус победы.

Заблокировать все буквы.

Если режим — «с компьютером», обновить статистику на сервере (Победа).

Если НЕТ:

Увеличить счётчик ошибок на 1.

Обновить изображение виселицы в зависимости от количества ошибок.

Проверить, достигнуто ли максимальное число ошибок (11):

Если ДА → игрок проиграл:

Установить статус поражения.

Заблокировать все буквы.

Если режим — «с компьютером», обновить статистику на сервере (Поражение).

Если НЕТ → показать оставшееся количество попыток.

```
&НаКлиенте
Процедура ОбработатьНажатуюБукву(Буква)
    // Добавить в историю
    Если ПустаяСтрока(ИспользованныеБуквы) Тогда
        ИспользованныеБуквы = Буква;
    Иначе
        ИспользованныеБуквы = ИспользованныеБуквы + ", "
+ Буква;
    КонецЕсли;
    Если Найти(ВРег(ЗагаданноеСлово), ВРег(Буква)) > 0
Тогда
        ОткрытьБукву(Буква);
        Если Найти(ПолеСлово, "_") = 0 Тогда // Все буквы
открыты!
            СтатусИгры = "Поздравляем! Вы выиграли!
Слово: " + ЗагаданноеСлово;
            ЗаблокироватьВсеБуквы();
            Если РежимИгры = "Компьютер" Тогда
          ОбновитьСтатистикуНаСервере(Игрок, "Победа");
            КонецЕсли;
        КонецЕсли;
    Иначе
        КоличествоОшибок = КоличествоОшибок + 1;
        ОбновитьКартинкуВиселицы();
        Если КоличествоОшибок >= 11 Тогда
            СтатусИгры = "Вы проиграли! Загаданное
слово: " + ЗагаданноеСлово;
            ЗаблокироватьВсеБуквы();
            Если РежимИгры = "Компьютер" Тогда
       ОбновитьСтатистикуНаСервере(Игрок, "Поражение");
```

```
            КонецЕсли;
        Иначе
            СтатусИгры = "Ошибка! Осталось попыток: " +
(11 - КоличествоОшибок);
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
```

14. Процедура ОткрытьБукву(Буква)

Назначение

Обновляет визуальное поле слова, раскрывая все вхождения указанной буквы.

Алгоритм

1. Создать пустую строку для нового состояния поля.

2. Пройти по каждой позиции в загаданном слове.

Для каждой позиции:

Определить позицию соответствующего символа в текущем ПолеСлово (формат: |_| → каждая буква на чётной позиции).

Если текущая буква в загаданном слове совпадает (регистронезависимо) с переданной — добавить её в новую строку.

Иначе — оставить уже существующий символ (букву или подчёркивание).

Добавить завершающую вертикальную черту |.

3. Присвоить новую строку реквизиту ПолеСлово.

```
&НаКлиенте
Процедура ОткрытьБукву(Буква)
    НоваяСтрока = "";
    Для i = 1 По СтрДлина(ЗагаданноеСлово) Цикл
// В формате "|_|A|", каждая буква находится на позиции
(i-1)*2 + 2
        ПозицияВПоле = (i - 1) * 2 + 2;
```

```
        ТекущийСимвол = Сред(ПолеСлово, ПозицияВПоле,
1); // То, что уже открыто
        // Сравниваем без учёта регистра
        Если ВРег(Сред(ЗагаданноеСлово, i, 1)) =
ВРег(Буква) Тогда
            НоваяСтрока = НоваяСтрока + "|" +
Сред(ЗагаданноеСлово, i, 1);
        Иначе
            НоваяСтрока = НоваяСтрока + "|" +
ТекущийСимвол;
        КонецЕсли;
    КонецЦикла;
    НоваяСтрока = НоваяСтрока + "|"; // Закрывающая черта
    ПолеСлово = НоваяСтрока;
КонецПроцедуры
```

15. Процедура ОбновитьКартинкуВиселицы() и функция ПолучитьКартинкуВиселицы(Номер)

Назначение

Отображает соответствующую стадию виселицы в зависимости от количества ошибок.

Алгоритм

Процедура ОбновитьКартинкуВиселицы():

1. Вызвать функцию ПолучитьКартинкуВиселицы, передав текущее КоличествоОшибок.

2. Присвоить возвращённое изображение реквизиту КартинкаВиселицы.

Функция ПолучитьКартинкуВиселицы(Номер):

1. На основании входного номера (от 1 до 11) выбрать соответствующий элемент-картинку на форме (Картинка1, Картинка2, ..., Картинка11).

2. Вернуть его свойство .Картинка.

3. Если номер вне диапазона — вернуть Неопределено.

```
&НаКлиенте
Функция ПолучитьКартинкуВиселицы(Номер)
    // Предполагается, что на форме есть 11 скрытых
элементов Картинка1...Картинка11
    Выборка = Номер;
    Если Выборка = 1 Тогда
        Возврат Элементы.Картинка1.Картинка;
    ИначеЕсли Выборка = 2 Тогда
        Возврат Элементы.Картинка2.Картинка;
    // ... и так до 11
    ИначеЕсли Номер = 11 Тогда
        Возврат Элементы.Картинка11.Картинка;
    Иначе
        Возврат Неопределено;
    КонецЕсли;
КонецФункции
```

```
&НаКлиенте
Процедура ОбновитьКартинкуВиселицы()
КартинкаВиселицы =
ПолучитьКартинкуВиселицы(КоличествоОшибок);
КонецПроцедуры
```

16. Серверные процедуры

Функция ПолучитьСлучайноеСлово()

Алгоритм

1. Выполнить запрос к справочнику «Слова».

2. Собрать все наименования в массив.

Если массив не пуст:

a. Создать генератор случайных чисел.

b. Получить случайный индекс от 0 до (количество − 1).

c. Вернуть слово с этим индексом.

Иначе вернуть пустую строку.

Процедура ОбновитьСтатистикуНаСервере(ССылкаНаИгрока, Результат)

Алгоритм

1. Если ссылка на игрока не указана — выйти.
2. Получить объект игрока по ссылке.
3. В зависимости от результата:
   a. При «Победа» — увеличить счётчик побед.
   b. При «Поражение» — увеличить счётчик поражений.
4. Пересчитать рейтинг как победы минус поражения.
5. Записать изменения в базу.

```
&НаСервере
Функция ПолучитьСлучайноеСлово()
    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |    Слова.Наименование КАК Наименование
    |ИЗ
    |    Справочник.Слова КАК Слова";
    Результат = Запрос.Выполнить();
    Выборка = Результат.Выбрать();
    Слова = Новый Массив;
    Пока Выборка.Следующий() Цикл
        Слова.Добавить(Выборка.Наименование);
    КонецЦикла;
    Если Слова.Количество() > 0 Тогда
```

```
        ГСЧ = Новый ГенераторСлучайныхЧисел;
        СлучайныйИндекс = ГСЧ.СлучайноеЧисло(0,
Слова.Количество() - 1);
        Возврат Слова[СлучайныйИндекс];
    Иначе
        Возврат "";
    КонецЕсли;
КонецФункции
```

```
&НаСервере
Процедура ОбновитьСтатистикуНаСервере(ССылкаНаИгрока,
Результат)
    Если ССылкаНаИгрока = Неопределено Тогда
        Возврат;
    КонецЕсли;
    ИгрокОбъект = ССылкаНаИгрока.ПолучитьОбъект();
    Если Результат = "Победа" Тогда
        ИгрокОбъект.Побед = ИгрокОбъект.Побед + 1;
    ИначеЕсли Результат = "Поражение" Тогда
     ИгрокОбъект.Поражений = ИгрокОбъект.Поражений + 1;
    КонецЕсли;
    ИгрокОбъект.Рейтинг = ИгрокОбъект.Побед -
ИгрокОбъект.Поражений;
    ИгрокОбъект.Записать();
КонецПроцедуры
```

Создание игры «Города» в среде 1С:Предприятие

Реализация игры студента группы ИС-31 Чащевого Дмитрия

Цель: разработать интерактивную игру «Города» в среде 1С:Предприятие, используя справочник городов, реализовать логику игры с искусственным интеллектом, включая проверку правил, валидацию ввода и обработку игровых состояний.

Задачи:

1. Создать внешнюю обработку с формой для игры.
2. Реализовать логику хода игрока и компьютера.
3. Обеспечить проверку введённого города:
    - наличие в справочнике,
    - совпадение начальной буквы с последней буквой предыдущего города,
    - отсутствие в списке уже использованных.
4. Организовать клиент-серверное взаимодействие для обработки запросов к справочнику.
5. Протестировать игру и оценить её работоспособность.

Подводящие вопросы для активизации мышления:

- Какие правила игры «Города» вы знаете? Городá — игра для нескольких (двух или более) человек, в которой каждый участник в свою очередь называет реально существующий в данный момент времени город любой существующей страны, название которого начинается на ту букву, которой оканчивается название предыдущего города.
- Как можно хранить список городов? (справочник)
- Как можно быстро заполнить справочник (импортировать список, например, с файла электронных таблиц).
- Как определить, что введённый город начинается на нужную букву?
- Как отследить, что город уже был использован?
- Что делается, если название города заканчивается на ь, ы, ъ?

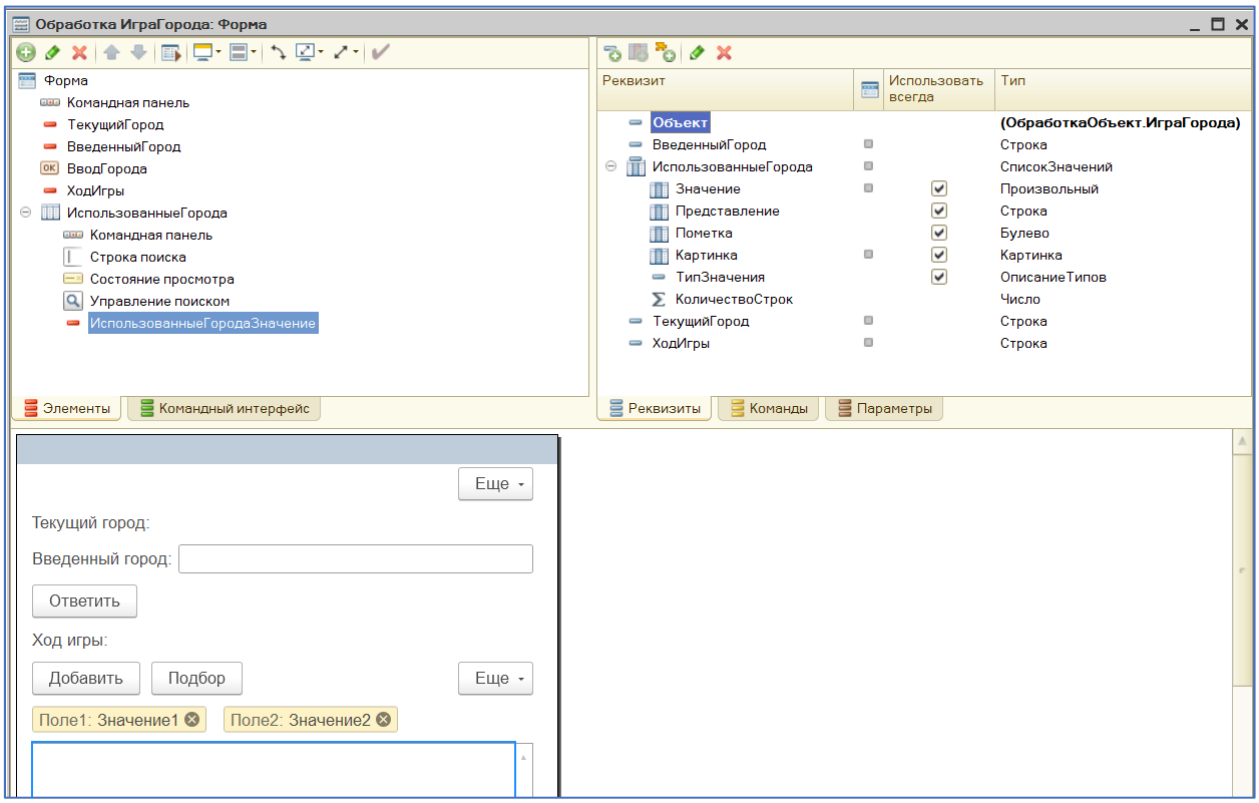
- Почему логику проверки нужно выполнять на сервере?
- Как компьютер может выбрать подходящий город из справочника?

Разработка интерфейса

Справочник Города должен содержать список городов в поле Наименование.

Форма содержит реквизиты:

- ТекущийГород (Строка) — хранит название города, на который нужно отвечать.
- ВведенныйГород (Строка) — поле ввода для игрока.
- ХодИгры (Строка, многострочный) — поле для отображения истории хода игры.
- ИспользованныеГорода: реквизит формы типа СписокЗначений, хранящий уже названные города.



Разработка программы

ПриОткрытии	Инициализация игры	Вызывает серверную процедуру для старта игры.

НачальныйХодКомпьютераНаСервере	Выбор начального города	Запрашивает все города, выбирает случайный, добавляет в список использованных, обновляет форму.
ВводГорода	Обработка хода игрока	Проверяет ввод: пустота, существование в справочнике, правильная буква, уникальность. Если всё ок — передаёт ход компьютеру.
ПроверитьГородВСправочнике	Проверка существования города	Запрос к справочнику по имени. Возвращает Истина, если найден.
ПолучитьПоследнююБукву	Определение ключевой буквы	Извлекает последнюю букву, учитывает "ь", "ъ", "ы".
ХодКомпьютераНаСервере	Ход ИИ	Ищет подходящий город (по начальной букве и отсутствию в списке), добавляет его, обновляет интерфейс.

1. Процедура ПриОткрытии (Клиентская)

Назначение

Инициирует начало игры, вызывая ход компьютера при открытии формы.

Компьютер случайным образом выбирает первый город.

Алгоритм

Вызвать серверную процедуру НачальныйХодКомпьютераНаСервере.

```
&НаКлиенте

Процедура ПриОткрытии(Отказ)

    // При открытии формы вызывается серверная процедура для начального хода компьютера

    НачальныйХодКомпьютераНаСервере();

КонецПроцедуры
```

2. Процедура НачальныйХодКомпьютераНаСервере (Серверная)

Назначение

Выполняет начальный ход компьютера: загружает список городов, выбирает случайный город из справочника, инициализирует список использованных городов и обновляет реквизиты формы.

Алгоритм

1. Выполнить SQL-запрос к справочнику Города, получить все наименования.
2. Проверить, есть ли хотя бы один город в справочнике.
3. Если справочник пуст, вывести сообщение и завершить выполнение.
4. Сгенерировать случайный индекс в пределах количества городов.
5. Выбрать город по этому индексу.
6. Создать список ИспользованныеГорода и добавить туда выбранный город.
7. Обновить реквизиты формы ТекущийГород и ХодИгры.

```
&НаСервере
Процедура НачальныйХодКомпьютераНаСервере()
    // Создаем новый объект запроса
    Запрос = Новый Запрос;
    // Устанавливаем текст запроса для получения всех
наименований городов
    Запрос.Текст = "
        |ВЫБРАТЬ
        |   Города.Наименование
        |ИЗ
        |   Справочник.Города КАК Города";
    // Выполняем запрос и выгружаем результат в таблицу
    Результат = Запрос.Выполнить();
    ТаблицаГородов = Результат.Выгрузить();
    // Проверяем, есть ли города в таблице
    Если ТаблицаГородов.Количество() = 0 Тогда
        // Если нет, выводим сообщение об ошибке
        Сообщить("Справочник «Города» пуст!");
        Возврат; // Прерываем выполнение процедуры
    КонецЕсли;
    // Создаем генератор случайных чисел
    Генератор = Новый ГенераторСлучайныхЧисел();
// Генерируем случайный индекс от 0 до (количество городов - 1)
```

```
    Индекс = Генератор.СлучайноеЧисло(0,
ТаблицаГородов.Количество() - 1);
    // Выбираем город по случайному индексу
    ТекущийГород = ТаблицаГородов[Индекс].Наименование;
    // Инициализируем список использованных городов
    ИспользованныеГорода = Новый СписокЗначений;
    // Добавляем первый выбранный город в список
    ИспользованныеГорода.Добавить(ТекущийГород);
    // Обновляем реквизит формы с текущим городом
    ЭтаФорма.ТекущийГород = ТекущийГород;
    // Обновляем реквизит формы с историей хода, указывая, что
это ход компьютера
    ЭтаФорма.ХодИгры = "Компьютер: " + ТекущийГород;
КонецПроцедуры
```

3. Процедура ВводГорода (Клиентская)

Назначение

Обрабатывает ввод игрока, проверяет корректность введенного города (наличие в справочнике, правильность начальной буквы, отсутствие в списке использованных) и передает ход компьютеру.

Алгоритм

1. Получить значение из поля ВведенныйГород.
2. Проверить, не пустое ли значение.
3. Проверить, существует ли такой город в справочнике, вызвав ПроверитьГородВСправочнике.
4. Проверить, начинается ли введенный город на последнюю букву текущего города, вызвав ПолучитьПоследнююБукву.
5. Проверить, не был ли этот город уже назван.
6. Если все проверки пройдены, добавить город в список использованных, обновить историю хода и вызвать ХодКомпьютераНаСервере.

```
&НаКлиенте
Процедура ВводГорода(Команда)
    // Получаем значение из поля ввода
```

```
    ВведенныйГород = ЭтаФорма.ВведенныйГород;
    // Проверяем, не пустая ли строка
    Если ПустаяСтрока(ВведенныйГород) Тогда
 Сообщить("Введите название города!"); // Выводим сообщение
        Возврат; // Прерываем выполнение
    КонецЕсли;
    // Проверяем, есть ли город в справочнике
    ЕстьВСправочнике =
ПроверитьГородВСправочнике(ВведенныйГород);
    Если Не ЕстьВСправочнике Тогда
        Сообщить("Такого города нет в справочнике!"); // Выводим
сообщение
        Возврат; // Прерываем выполнение
    КонецЕсли;
    // Получаем последнюю букву текущего города
    ПоследняяБуква =
ПолучитьПоследнююБукву(ЭтаФорма.ТекущийГород);
    // Получаем первую букву введенного города
    ПерваяБуква = Нрег(Лев(ВведенныйГород, 1)); // Нрег -
приведение к нижнему регистру
    // Проверяем, совпадает ли первая буква введенного с
последней буквой текущего
    Если ПерваяБуква <> ПоследняяБуква Тогда
        Сообщить("Город должен начинаться на букву «" +
ПоследняяБуква + "»!"); // Выводим сообщение
        Возврат; // Прерываем выполнение
    КонецЕсли;
    // Проверяем, не был ли город уже назван
    Если
ИспользованныеГорода.НайтиПоЗначению(НРег(ВведенныйГород)) <>
Неопределено Тогда
        Сообщить("Этот город уже был назван!"); // Выводим
сообщение
        Возврат; // Прерываем выполнение
    КонецЕсли;
```

```
    // Если все проверки пройдены, добавляем город в список
использованных
    ИспользованныеГорода.Добавить(ВведенныйГород);
    // Обновляем историю хода, указывая, что это ход игрока
    ЭтаФорма.ХодИгры = "Игрок: " + ЭтаФорма.ВведенныйГород;
    // Обновляем текущий город
    ЭтаФорма.ТекущийГород = ВведенныйГород;
    // Передаем ход компьютеру
    ХодКомпьютераНаСервере();
КонецПроцедуры
```

4. Функция ПроверитьГородВСправочнике (Серверная)

Назначение

Проверяет, существует ли введенный пользователем город в справочнике Города.

Алгоритм

1. Выполнить SQL-запрос с параметром, проверяющий существование города в справочнике.
2. Вернуть Истина, если запись найдена, и Ложь в противном случае.

```
&НаСервере
Функция ПроверитьГородВСправочнике(Город)
    // Создаем новый объект запроса
    Запрос = Новый Запрос;
    // Устанавливаем текст запроса с параметром &Город
    Запрос.Текст = "
        |ВЫБРАТЬ
        |   1 // Выбираем любое значение, если условие
выполняется
        |ИЗ
        |   Справочник.Города КАК Города
        |ГДЕ
        |   Города.Наименование = &Город"; // Используем
параметр
    // Устанавливаем значение параметра
```

```
    Запрос.УстановитьПараметр("Город", Город);
    // Выполняем запрос
    Результат = Запрос.Выполнить();
    // Возвращаем Истина, если результат не пустой (город
найден), иначе Ложь
    Возврат Не Результат.Пустой();
КонецФункции
```

5. Функция ПолучитьПоследнююБукву (Серверная)

Назначение

Возвращает последнюю значимую букву названия города. Учитывает мягкий и твердый знаки, а также букву "ы", считая их незначимыми в контексте игры "Города".

Алгоритм

1. Получить последний символ строки.
2. Если это "ь", "ъ" или "ы", вернуть предпоследнюю букву.
3. Иначе, вернуть последнюю букву.

```
&НаСервере
Функция ПолучитьПоследнююБукву(Город)
    // Получаем последний символ строки
    ПоследняяСимвол = Прав(Город, 1);
    // Проверяем, является ли последний символ незначимой буквой
    Если ПоследняяСимвол = "ь" Или ПоследняяСимвол = "ъ" Или
ПоследняяСимвол = "ы" Тогда
        // Возвращаем предпоследнюю букву
        Возврат Сред(Город, СтрДлина(Город) - 1, 1);
    Иначе
        // Возвращаем последнюю букву
        Возврат ПоследняяСимвол;
    КонецЕсли;
КонецФункции
```

6. Процедура ХодКомпьютераНаСервере (Серверная)

Назначение

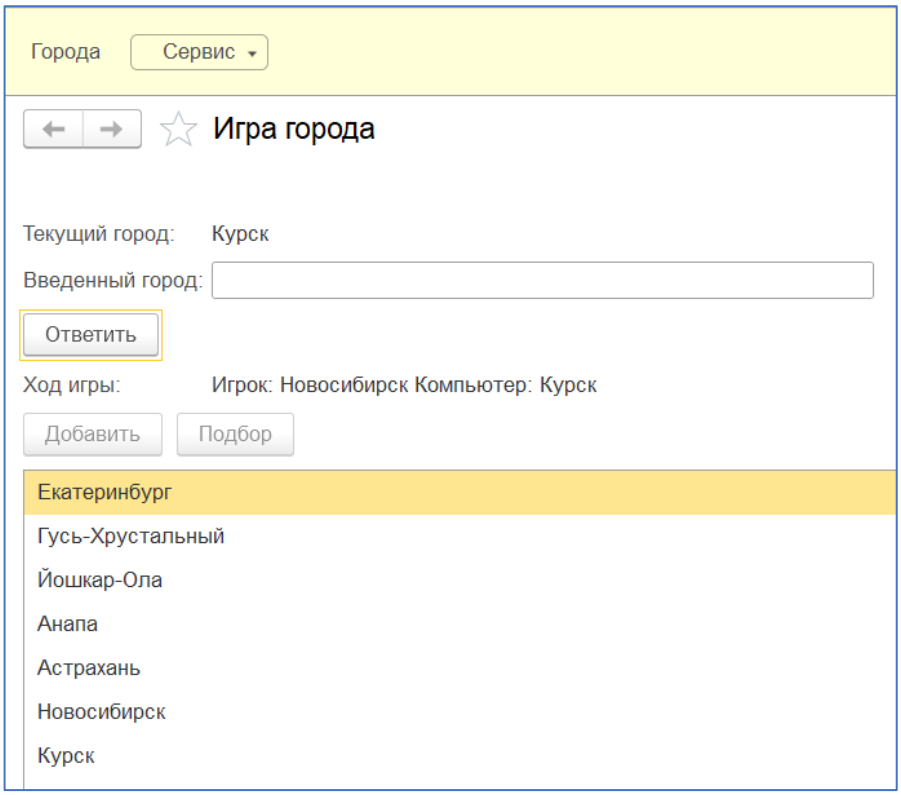
Выполняет ход компьютера: находит подходящий город, начинающийся на нужную букву и не использовавшийся ранее, и обновляет состояние игры. Если подходящих городов нет, объявляет победу игрока.

Алгоритм

1. Выполнить SQL-запрос, выбирающий города, начинающиеся на нужную букву и не находящиеся в списке ИспользованныеГорода.
2. Проверить, есть ли подходящие города.
3. Если нет, обновить историю хода, сообщив о победе игрока.
4. Если есть, выбрать случайный город из результата.
5. Добавить выбранный город в список использованных.
6. Обновить реквизиты формы ТекущийГород, ХодИгры и ВведенныйГород.

```
&НаСервере
Процедура ХодКомпьютераНаСервере()
    // Создаем новый объект запроса
    Запрос = Новый Запрос;
    // Устанавливаем текст запроса с параметрами
    Запрос.Текст = "
        |ВЫБРАТЬ
        |   Города.Наименование
        |ИЗ
        |   Справочник.Города КАК Города
        |ГДЕ
        |   НЕ Города.Наименование В (&ИспользованныеГорода) //
Исключаем использованные города
        |   И ПОДСТРОКА(Города.Наименование, 1, 1) =
&ПерваяБуква"; // Фильтр по начальной букве
    // Получаем нужную начальную букву (это последняя буква
предыдущего хода)
    ПоследняяБуква =
ПолучитьПоследнююБукву(ЭтаФорма.ТекущийГород);
    // Устанавливаем параметр запроса для начальной буквы
    Запрос.УстановитьПараметр("ПерваяБуква", ПоследняяБуква);
```

```
    // Выгружаем значения из списка использованных городов для
параметра
    Запрос.УстановитьПараметр("ИспользованныеГорода",
ИспользованныеГорода.ВыгрузитьЗначения());
    // Выполняем запрос
    Результат = Запрос.Выполнить();
    ТаблицаГородов = Результат.Выгрузить(); // Выгружаем
результат
    // Проверяем, есть ли подходящие города
    Если ТаблицаГородов.Количество() = 0 Тогда
        // Если нет, обновляем историю хода, объявляя победу
игрока
        ЭтаФорма.ХодИгры = ЭтаФорма.ХодИгры + Символы.ПС +
"Компьютер не может сделать ход! Вы победили!";
        Возврат; // Прерываем выполнение
    КонецЕсли;
    // Создаем генератор случайных чисел
    Генератор = Новый ГенераторСлучайныхЧисел();
    // Генерируем случайный индекс для выбора города из
результата
    Индекс = Генератор.СлучайноеЧисло(0,
ТаблицаГородов.Количество() - 1);
    // Выбираем новый город
    НовыйГород = ТаблицаГородов[Индекс].Наименование;
    // Добавляем выбранный город в список использованных
    ИспользованныеГорода.Добавить(НовыйГород);
    // Обновляем историю хода, добавляя ход компьютера
    ЭтаФорма.ХодИгры = ЭтаФорма.ХодИгры + Символы.ПС +
"Компьютер: " + НовыйГород;
    // Обновляем текущий город
    ЭтаФорма.ТекущийГород = НовыйГород;
    // Очищаем поле ввода игрока для следующего хода
    ЭтаФорма.ВведенныйГород = "";
КонецПроцедуры
```



Дополнительное задание

В перспективе хотелось бы получать фотографию города, который выбрал пользователь или компьютер. Нужен вариант загрузки и хранения фотографий.

Индивидуальные задания на разработку игр

1. Орёл и решка (*)
2. Игровой автомат (*)
3. Камень, ножницы, бумага (*)
4. Пятнадцать (***)
5. Карточная игра УНО (***)
6. Морской бой (****)
7. Змейка (****)
8. Сапёр (****)

Примеры реализации

Камень, ножницы, бумага

(реализация студента гр. ИС-31 Вдовина Никиты)

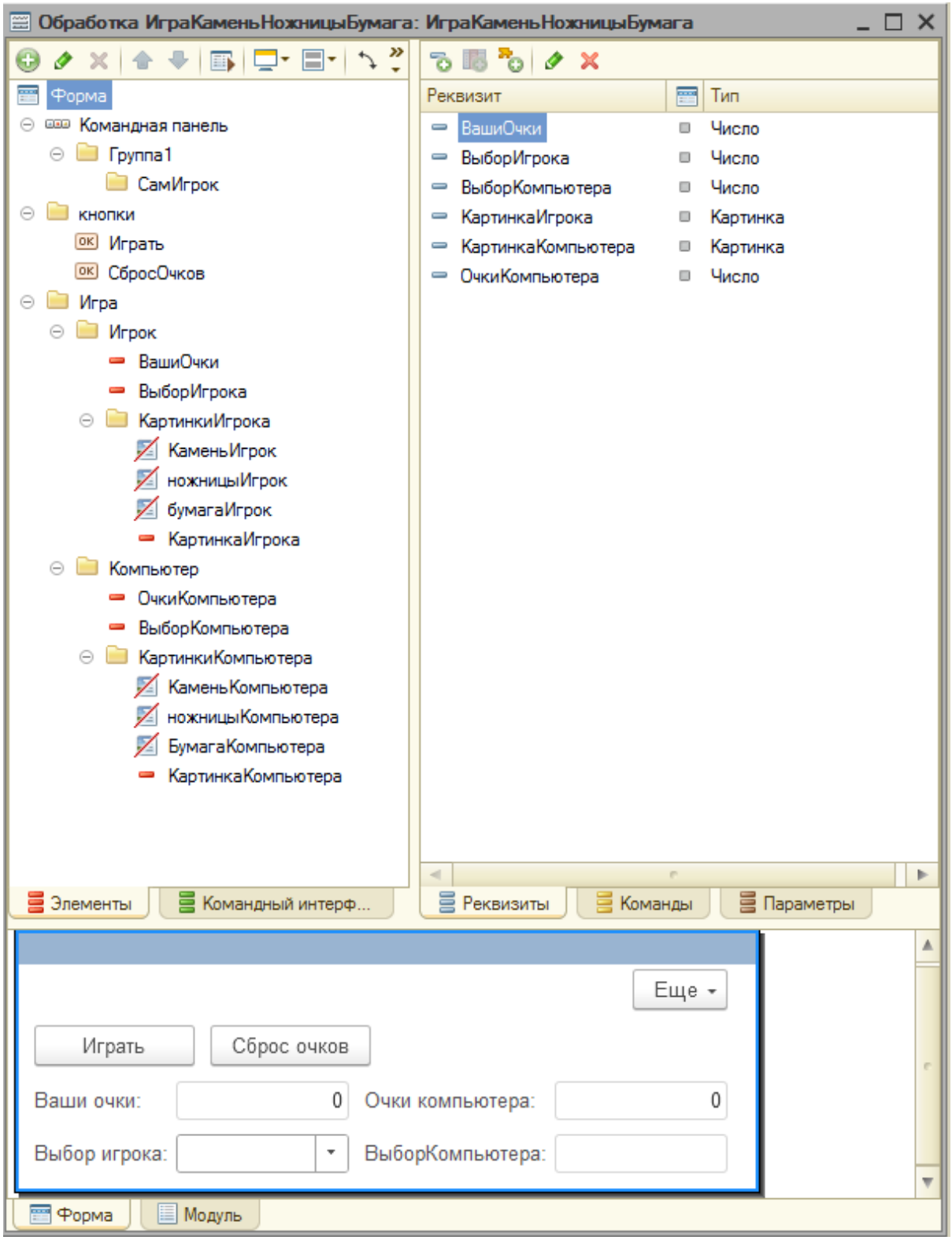
Правила игры

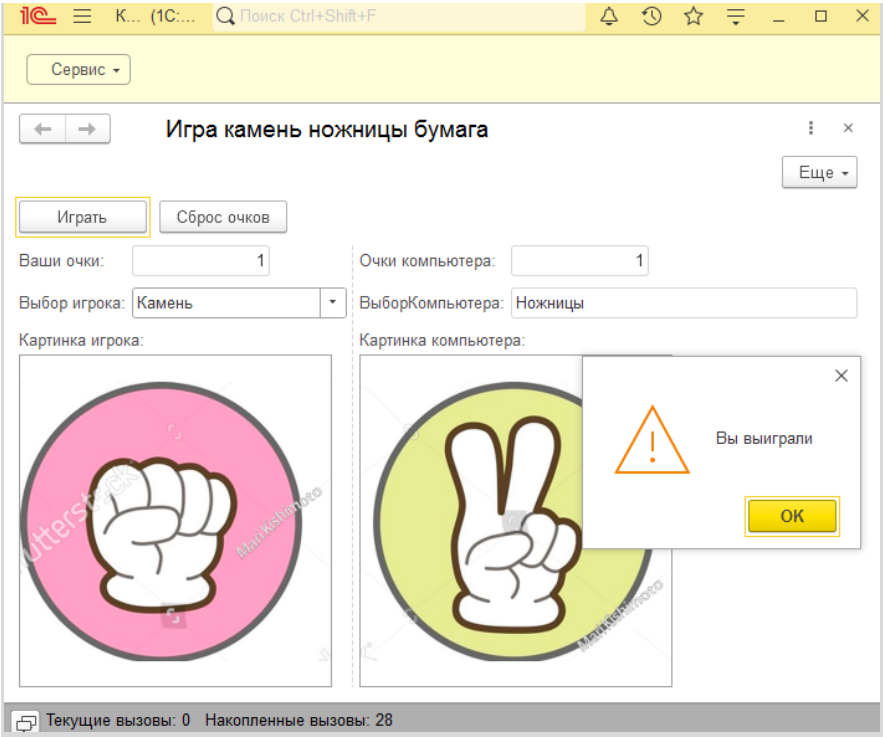
Игроки считают вместе вслух «Камень… Ножницы… Бумага… Раз… Два… Три», одновременно качая кулаками. На счёт «Три» они одновременно показывают при помощи руки один из трёх знаков: камень, ножницы или бумагу.

Победитель определяется по следующим правилам:

- Бумага побеждает камень («бумага обёртывает камень»).
- Камень побеждает ножницы («камень затупляет ножницы»).
- Ножницы побеждают бумагу («ножницы разрезают бумагу»)[1].

Если игроки показали одинаковый знак, то засчитывается ничья[1] и игра переигрывается.


Обработка ИграКаменьНожницыБумага: ИграКаменьНожницыБумага
Форма
Командная панель
Группа1
СамИгрок
кнопки
Играть
СбросОчков
Игра
Игрок
ВашиОчки
ВыборИгрока
КартинкиИгрока
КаменьИгрок
ножницыИгрок
бумагаИгрок
КартинкаИгрока
Компьютер
ОчкиКомпьютера
ВыборКомпьютера
КартинкиКомпьютера
КаменьКомпьютера
ножницыКомпьютера
БумагаКомпьютера
КартинкаКомпьютера
Реквизит
Тип
ВашиОчки
Число
ВыборИгрока
Число
ВыборКомпьютера
Число
КартинкаИгрока
Картинка
КартинкаКомпьютера
Картинка
ОчкиКомпьютера
Число
Элементы
Командный интерф...
Реквизиты
Команды
Параметры
Еще
Играть
Сброс очков
Ваши очки:
0
Очки компьютера:
0
Выбор игрока:
ВыборКомпьютера:
Форма
Модуль




```
#Область Игры
// Обработчик нажатия кнопки "Играть"
// Реализует логику одного раунда игры "Камень, ножницы, бумага"
&НаКлиенте
Процедура Играть(Команда)
// Создаём генератор случайных чисел для выбора хода компьютера
    ГСЧ = Новый ГенераторСлучайныхЧисел;
// Проверяем, сделал ли игрок выбор (1 — камень, 2 — ножницы, 3 — бумага)
    Если ЭтотОбъект.ВыборИгрока = 0 Тогда
        ПредупреждениеАсинх("Сделай выбор, чтобы начать игру");
        Возврат; // Прерываем выполнение, если выбор не сделан
    КонецЕсли;
    // Компьютер случайно выбирает число от 1 до 3
```

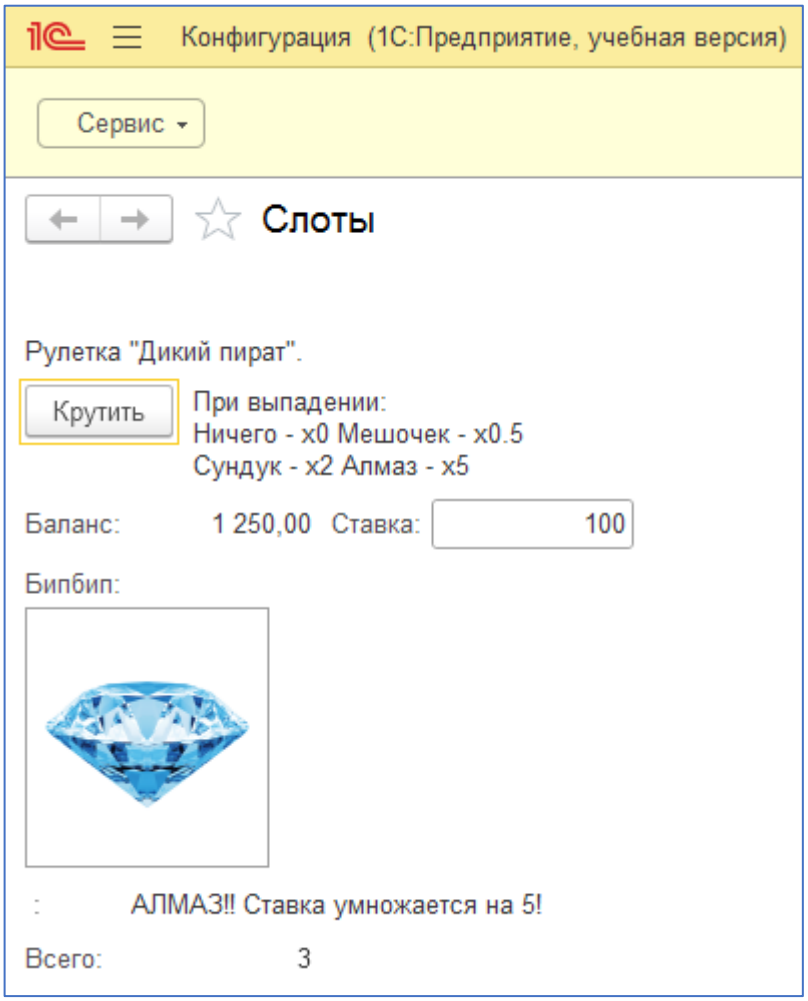
```
    ЭтотОбъект.ВыборКомпьютера = ГСЧ.СлучайноеЧисло(1,
3);
    // Получаем картинку, соответствующую выбору
компьютера, и отображаем её
    ЭтотОбъект.КартинкаКомпьютера =
ПолучитьКартинкуКомпьютера(ЭтотОбъект.ВыборКомпьютера);
    Элементы.КартинкаКомпьютера.Видимость = Истина;
    // Сравниваем выборы игрока и компьютера
    Если ЭтотОбъект.ВыборИгрока =
ЭтотОбъект.ВыборКомпьютера Тогда
        // Ничья
        ПредупреждениеАсинх("Ничья");
    Иначе
        // Проверяем выигрышные комбинации для игрока:
        // Камень (1) побеждает ножницы (2)
        // Ножницы (2) побеждают бумагу (3)
        // Бумага (3) побеждает камень (1)
        Если (ЭтотОбъект.ВыборИгрока = 1 И
ЭтотОбъект.ВыборКомпьютера = 2) Или
           (ЭтотОбъект.ВыборИгрока = 2 И
ЭтотОбъект.ВыборКомпьютера = 3) Или
           (ЭтотОбъект.ВыборИгрока = 3 И
ЭтотОбъект.ВыборКомпьютера = 1) Тогда
            ПредупреждениеАсинх("Вы выиграли");
            ЭтотОбъект.ВашиОчки = ЭтотОбъект.ВашиОчки +
1;
        Иначе
            // Во всех остальных случаях — игрок
проиграл
            ПредупреждениеАсинх("Вы проиграли");
```

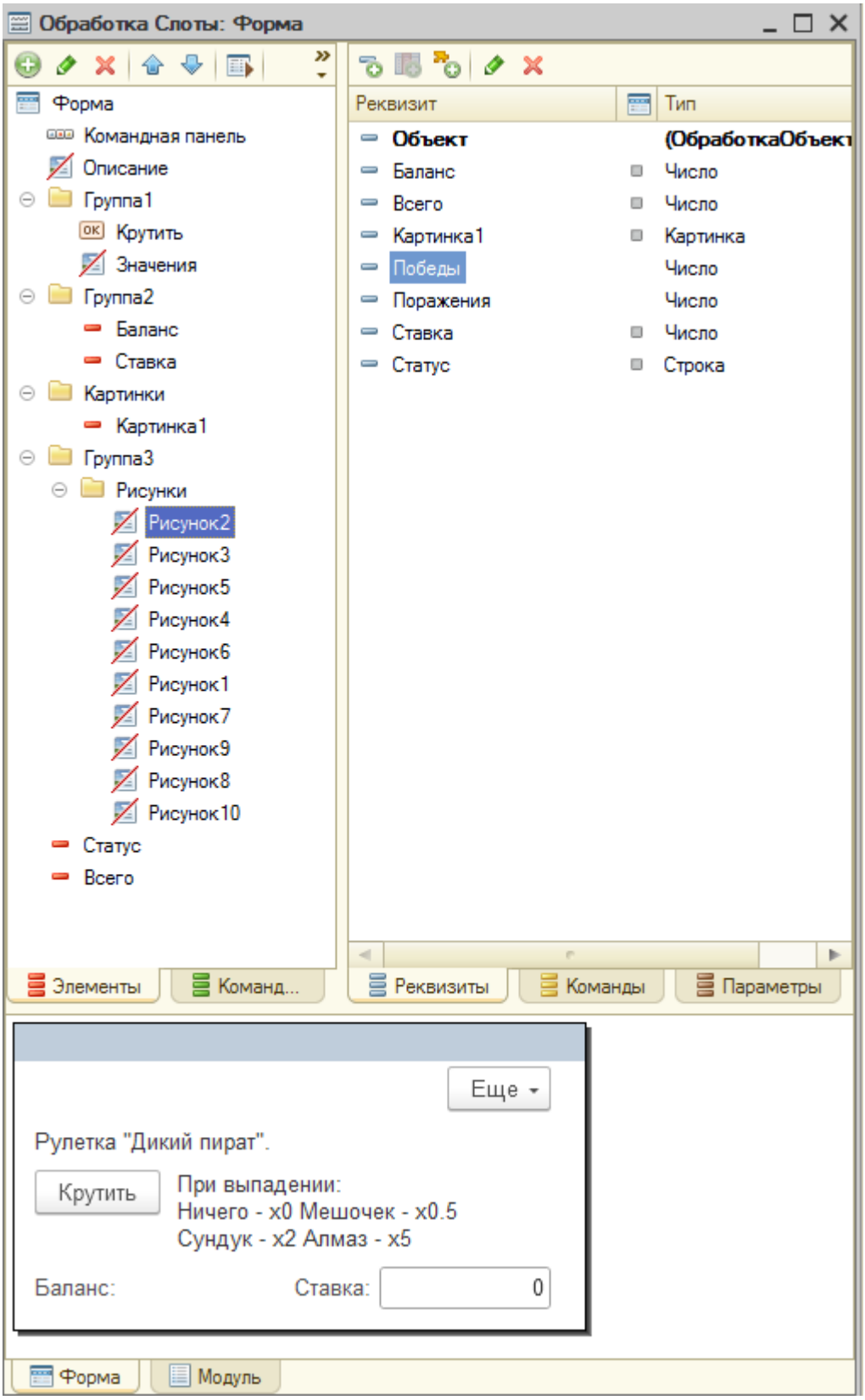
```
            ЭтотОбъект.ОчкиКомпьютера                    =
ЭтотОбъект.ОчкиКомпьютера + 1;
        КонецЕсли;
    КонецЕсли;
КонецПроцедуры
// Обработчик кнопки "Сброс очков"
&НаКлиенте
Процедура СбросОчков(Команда)
    // Обнуляем счёт
    ЭтотОбъект.ОчкиКомпьютера = 0;
    ЭтотОбъект.ВашиОчки = 0;
    // Скрываем изображения выбора игрока и компьютера
    Элементы.КартинкаИгрока.Видимость = Ложь;
    Элементы.КартинкаКомпьютера.Видимость = Ложь;
КонецПроцедуры

// Обработчик изменения выбора игрока (например, при
выборе переключателя)
&НаКлиенте
Процедура ВыборИгрокаПриИзменении(Элемент)
    // Получаем картинку, соответствующую текущему
выбору игрока
    ЭтотОбъект.КартинкаИгрока                    =
ПолучитьКартинкуИгрока(ЭтотОбъект.ВыборИгрока);
    // Отображаем картинку на форме
    Элементы.КартинкаИгрока.Видимость = Истина;
КонецПроцедуры

// Вспомогательная функция: возвращает картинку игрока
по числовому коду выбора
```

```
&НаКлиенте
Функция ПолучитьКартинкуИгрока(Число)
     Если Число = 1 Тогда
         Возврат Элементы.КаменьИгрок.Картинка;        //
Камень
     ИначеЕсли Число = 2 Тогда
         Возврат Элементы.ножницыИгрок.Картинка;       //
Ножницы
     ИначеЕсли Число = 3 Тогда
         Возврат Элементы.бумагаИгрок.Картинка;        //
Бумага
     Иначе
         Возврат Неопределено; // На случай некорректного
значения
     КонецЕсли;
КонецФункции

//   Вспомогательная   функция:   возвращает   картинку
компьютера по числовому коду выбора
&НаКлиенте
Функция ПолучитьКартинкуКомпьютера(Число)
     Если Число = 1 Тогда
Возврат  Элементы.КаменьКомпьютера.Картинка;           //
Камень
     ИначеЕсли Число = 2 Тогда
     Возврат Элементы.ножницыКомпьютера.Картинка;      //
Ножницы
     ИначеЕсли Число = 3 Тогда
     Возврат Элементы.бумагаКомпьютера.Картинка;       //
Бумага
```

```
    Иначе
        Возврат Неопределено;
    КонецЕсли;
КонецФункции
#КонецОбласти
```

Игровой автомат






```
// Процедура вызывается при открытии формы
// Инициализирует начальное состояние игры
&НаКлиенте
Процедура ПриОткрытии(Отказ)
```

```
// Устанавливаем начальный баланс игрока (виртуальные деньги)
    ЭтотОбъект.Баланс = 1000;
// Скрываем элементы, которые появятся только после первого хода:
    Элементы.Картинка1.Видимость = Ложь;    // Выпавший символ
    Элементы.Статус.Видимость = Ложь;       // Сообщение о результате
Элементы.Всего.Видимость = Ложь;            // Счётчик количества игр
КонецПроцедуры
// Обработчик нажатия кнопки "Крутить" (основная игровая логика)
&НаКлиенте
Процедура Крутить(Команда)
    // Сразу вычитаем ставку из баланса (даже если игрок проиграет)
    ЭтотОбъект.Баланс = ЭтотОбъект.Баланс - ЭтотОбъект.Ставка;
    // Проверка: ставка не может превышать текущий баланс
    Если ЭтотОбъект.Ставка > ЭтотОбъект.Баланс Тогда
        Предупреждение("Ставка больше чем баланс!", 5, "Ошибка!");
        Возврат; // Прерываем выполнение
    КонецЕсли;
    // Проверка: ставка должна быть положительной
    Если ЭтотОбъект.Ставка <= 0 Тогда
```

```
        Предупреждение("Сделайте ставку!", 5, "Ошибка!");
        Возврат;
    КонецЕсли;
    // Генерируем случайное число от 1 до 10 — это определит результат
    ГСЧ = Новый ГенераторСлучайныхЧисел;
    Число = ГСЧ.СлучайноеЧисло(1, 10);
    // Показываем выпавшую картинку
    Элементы.Картинка1.Видимость = Истина;
    ЭтотОбъект.Картинка1 = ПолучитьКартинку(Число);
    // Определяем выигрыш в зависимости от выпавшего числа
    Если (Число = 1) Или (Число = 2) Или (Число = 3) Или (Число = 4) Или (Число = 5) Или (Число = 6) Тогда
        // Вероятность: 60% — ничего не выигрываем
        ЭтотОбъект.Статус = "Ничего не выпало! Блин! Ставка умножается на 0!";
        ЭтотОбъект.Баланс = ЭтотОбъект.Баланс + (ЭтотОбъект.Ставка * 0); // Ничего не возвращаем
    ИначеЕсли (Число = 7) Или (Число = 8) Тогда
        // Вероятность: 20% — частичный возврат
        ЭтотОбъект.Статус = "Выпал мешочек с деньгами! Ставка умножается на 0.5!";
        ЭтотОбъект.Баланс = ЭтотОбъект.Баланс + (ЭтотОбъект.Ставка * 0.5);
    ИначеЕсли Число = 9 Тогда
        // Вероятность: 10% — хороший выигрыш
        ЭтотОбъект.Статус = "Выпал сундук с сокровищами! Ставка умножается на 2!";
```

```
        ЭтотОбъект.Баланс    =    ЭтотОбъект.Баланс    +
(ЭтотОбъект.Ставка * 2);
    ИначеЕсли Число = 10 Тогда
        // Вероятность: 10% — джекпот!
        ЭтотОбъект.Статус = "АЛМАЗ!! Ставка умножается
на 5!";
        ЭтотОбъект.Баланс    =    ЭтотОбъект.Баланс    +
(ЭтотОбъект.Ставка * 5);
    Иначе
        // Этот блок технически недостижим, но оставлен
для полноты
    КонецЕсли;
    // Увеличиваем счётчик общего количества сыгранных
раундов
    ЭтотОбъект.Всего = ЭтотОбъект.Всего + 1;
    // Отображаем результат игроку
    Элементы.Статус.Видимость = Истина;    // Сообщение
о выигрыше/проигрыше
    Элементы.Всего.Видимость = Истина;    // Показываем
счётчик игр
КонецПроцедуры

// Вспомогательная функция: возвращает картинку по номеру
// Предполагается, что на форме есть элементы с именами:
Рисунок1, Рисунок2, ..., Рисунок10
&НаКлиенте
Функция ПолучитьКартинку(Число)
    // Формируем имя элемента динамически
    ИмяКартинки = "Рисунок" + Число;
```

```
    // Получаем картинку из соответствующего элемента
формы
    Возврат Элементы[ИмяКартинки].Картинка;
КонецФункции
```

Список использованных источников

1. Радченко М.Г. 1С Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приёмы/ М.Г. Радченко, Е.Ю. Хрусталёва. – М.: 1С-Паблишинг, 2019. – 965с.
2. Сотников А., Сайфутдинов В. 10 ступеней к программированию на платформе 1С: Предприятие 8.3. «Верный старт в 1С», www.work-1c.ru.
3. Низамутдинов Ильяс. Программировать в 1С за 11 шагов. Простой пошаговый курс обучения программированию в 1С для всех.